3-14-2014

A Quantification of the 3D Modeling Capabilities of the Kinect Fustion Algorithm

Jeremy M. Higbee

A QUANTIFICATION OF THE 3D MODELING CAPABILITIES OF THE KINECTFUSTION ALGORITHM

THESIS

Jeremy M. Higbee, Captain, USAF

AFIT-ENG-14-M-40

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

AFIT-ENG-14-M-40

A QUANTIFICATION OF THE 3D MODELING CAPABILITIES OF THE KINECTFUSTION ALGORITHM

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Jeremy M. Higbee, BS

Captain, USAF

March 2014

AFIT-ENG-14-M-40

A QUANTIFICATION OF THE 3D MODELING CAPABILITIES OF THE KINECTFUSTION ALGORITHM

Jeremy M. Higbee, BS
Captain, USAF

Approved:

/signed/	13 Mar 2014
Maj Brian G. Woolley (Chair)	Date
/signed/	13 Mar 2014
Peter J. Collins, PhD (Member)	Date
/signed/	12 Mar 2014
Kyle J. Kauffman, PhD (Member)	Date

AFIT-ENG-14-M-40

Abstract

In the field of three-dimensional modeling, we continually struggle to quantify how closely the resulting model matches the physical object being represented. When precision measurements are required, they are often left to high-end, industrial systems. The aim of this thesis is to quantify the level of precision that can be obtained from commodity systems such as the Microsoft Kinect paired with the KinectFusion algorithm. Although the Kinect alone is considered a noisy sensor, the KinectFusion algorithm has shown the ability to build detailed surface models through the aggregation of depth information taken from multiple perspectives. This work represents the first rigorous validation of the three-dimensional modeling capabilities of the KinectFusion algorithm. One experiment is performed to measure the effects of key algorithm parameters such as resolution and range, while another is performed to measure the lower bounds at which objects can be detected and accurately modeled. The first experiment found that the KinectFusion algorithm reduced the uncertainty of the Kinect sensor alone from 10 mm to just 1.8 mm. Furthermore, the results of the second experiment demonstrate that the KinectFusion algorithm can detect surface deviations as little as 1.3 mm, but cannot accurately measure the deviation. Such results form an initial quantification of the KinectFusion algorithm, thus providing confidence about when and when not to utilize the KinectFusion algorithm for precision modeling. The hope is that this work will open the door for the algorithm to be used in real-world applications, such as alleviating the tedious visual surface inspections required for United States Air Force (USAF) aircraft.

To my family, without whom none of this would have been possible.

Acknowledgments

First and foremost, I'd like to thank my advisor, Major Brian Woolley, for all his support and guidance through this undertaking. Thank you for your patience, understanding, and the example you set everyday. We've come a long way from two guys brainstorming about three-dimensional modeling and the Kinect sensor.

I would also like to thank Rick Patton for all his hard work, support, and help in designing, building, and completing the experiments necessary for this work.

Additionally, thank you to the rest of the Advanced Navigation Technology (ANT) Center faculty and staff for all their support and resources in helping me complete this undertaking.

Finally, to my family, I can never repay the unconditional love and support that you have all provided during the last 18 months. I am the luckiest man in the world to have you guys!

Jeremy M. Higbee

Table of Contents

Page

Page

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
AMP	Accelerated Massive Parallelism
CAD	Computer Aided Design
CCD	Charge-Coupled Device
CMM	Coordinate-Measuring Machine
CMOS	Complementary metal oxide semiconductor
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
fps	frames per second
GPU	Graphics Processing Unit
ICP	Iterative Closest Point
IQR	Interquartile Range
IR	Infrared
LiDaR	Light Distance and Ranging
RAM	Random Access Memory
RGB	Red, Green, Blue
rpm	revolutions per minute
SDK	Software Development Kit
STL	Stereo Lithography
US	United States
USAF	United States Air Force
USB	Universal Serial Bus
VGA	Video Graphics Array
vpm	voxels per meter

A QUANTIFICATION OF THE 3D MODELING CAPABILITIES OF THE KINECTFUSTION ALGORITHM

I. Introduction

1.1 General Issues

Modeling and simulation techniques are employed to analyze systems where it is either infeasible or impossible to analyze the physical system. Specifically, three-dimensional modeling provides the capability to perform the analysis and inspection of complex physical objects. As computational capabilities have increased the ability to create models with greater detail has also increased. Today, three-dimensional sensors are used for a variety of purposes, such as post production inspections to determine the quality of individual parts on an assembly line or surveying large geographic areas via aircraft-mounted systems. However, such modeling systems are frequently limited in that they are often times expensive, overly large, and/or immobile. These drawbacks have kept such systems from realizing broader use, especially in the USAF.

The USAF maintains several technologically advanced aircraft that require extensive maintenance to retain their unique war fighting capabilities. At times, this maintenance requirement can exceed the available manpower, which reduces either the overall availability of an aircraft or its effectiveness. Currently, visual inspections are employed to detect defects and irregularities, which determine the amount of necessary maintenance. While the human eye and brain is quite adept for such visual inspections, utilizing maintenance personnel introduces certain limitations to the inspections. One such limitation is fatigue, caused by the sheer time required to search for small deviations on very large surfaces. Another drawback is that such inspections require a skilled crew to

identify only those anomalies that might affects its operational performance. Finally, when legitimate deviations are noted, the position information is recorded as an approximation in relation to major sections of the aircraft. While such approximations may be suitable for counting the number of defects or locating a defect for repair actions, it does not lend itself to detailed modeling and analysis actions that require more accuracy with regard to the defect position and characteristics.

Recent technological advances have increased the capabilities of both sensors and algorithms such that high quality results are available from cheaper, smaller, and more mobile devices. For instance, the Kinect sensor (originally released for the Microsoft Xbox 360 gaming console) provides both two-dimensional color imaging and a depth image, thus providing a three-dimensional system capable of tracking the skeletal movements of two humans simultaneously. Such capabilities were only dreams ten years ago. Remarkably, the Kinect sensor provides a suite of capabilities in an inexpensive and light weight package that can be employed as a hand-held device. Furthermore, such commodity equipment is easily replaced if broken. While the accuracy of the Kinect sensor itself is quite poor in comparison to industrial systems, the introduction of an innovative algorithm called *KinectFusion* [18] allows three-dimensional models to be built through the integration of sequential frames of depth information. The result is a system capable of constructing accurate three-dimensional models without large scale budgets. The natural question to ask is, how accurate are the models created by the KinectFusion algorithm?

1.2 Problem Statement

Since the release of the Kinect sensor in 2010, several different experiments have quantified the accuracy and precision of the device. Shortly thereafter, Izadi et al. [17] introduced the KinectFusion algorithm. Since its release, several efforts have been made to enhance the general performance of the algorithm [24], and to expand its three-dimensional modeling capabilities [10]. Surprisingly, no work has yet provided a quantification of the

algorithm's capabilities. Without such a quantification, the Kinect and KinectFusion will remain a toy system constricted to the video game arena, rather than a capability for rapidly acquiring detailed models for real-world applications.

1.3 Research Objectives

The aim of this work is to provide such a quantification, open the door for research into further improvements on the KinectFusion algorithm, and provide a metric for comparing this system against the larger, more expensive, industry standard modeling systems currently in use.

1.4 Research Focus

Given the research objectives, there are two areas that will be the focus for quantifying the accuracy and quality of models built using the KinectFusion algorithm and Kinect sensor. The first focus is the comparison of the three-dimensional model created by the KinectFusion algorithm to the original physical object being scanned. Such a comparison will provide insights into the accuracy of the KinectFusion algorithm when paired with the Kinect sensor. The second focus measures the lower limits at which an object can be detected and modeled by the system, shedding light on just how much the KinectFusion algorithm can improve the accuracy of measurements from a sensor — in this case, a Kinect sensor. Previously, Khoshelham and Elberink [20] showed that the Kinect sensor itself has an error of approximately 10 mm. The initial tests presented here indicate that the KinectFusion algorithm improves this so that surfaces as short as 1 mm can be detected.

These two focus areas lend themselves to the visual inspection task required for the maintenance of low observable aircraft. Thus, in support of such an objective, the aim of this work is to quantify the detection capabilities for surface defects (i.e. how small can something be before it can not be detected) and the accuracy of the detection capabilities (i.e. once detected, how closely does the detected defect match the actual physical defect).

The hope is that this will provide a basis for using a three-dimensional surface modeling approach that will replace the cumbersome human based approach currently in use.

1.5 Methodology

Two experiments will be performed to achieve the goals of this research. The first experiment, called the *model accuracy experiment*, will build three-dimensional models of a physical object using the KinectFusion algorithm. The resulting models will then be compared to a high-fidelity model taken from the same physical object. Since the high-fidelity model will be constructed by a Coordinate-Measuring Machine (CMM) with an accuracy of ±0.041 mm, this work will consider it to be *ground truth* for the physical object. Comparisons between the two models will utilize the fundamental Iterative Closest Point (ICP) [8] algorithm for alignment, followed by a calculation of the Euclidean distance between similar points within both models. These distances will form the basis for the quantification of the modeling capabilities of the KinectFusion algorithm.

The second experiment, known as the *anomaly detection experiment*, will build models of a static scene containing several objects of varying dimensions. The array of objects will provide a range of different height values above a planar surface, which will ultimate provide an excellent model for determining the lower bounds of the KinectFusion algorithm's modeling capabilities. In addition to simply determining what can be detected and what cannot be detected, this experiment will measure the precision at which very small objects can be detected and modeled. In fact, this experiment highlights the difference between detecting the presence of a 1 mm object, and accurately modeling the same object as a 1 mm object.

1.6 Assumptions and Limitations

The main limitation found in this work is that the only objects currently available and suitable for scanning with both the Kinect sensor and the CMM device are scale replica

models. Therefore the assumption of this thesis is that the accuracy and quality measured by scanning a scale model will also scale accordingly. However, there is no guarantee that the reported effects will scale as a simple linear factor. This work simply provides an initial quantification of the capabilities of the KinectFusion algorithm in the given scenario. Further quantifications will require additional experimentation that are beyond the scope of this work.

1.7 Implications

This work constitutes the first quantification of the improvements to three-dimensional modeling provided by the KinectFusion algorithm. Understanding the improvement provided by the algorithm creates the confidence that can open the door for utilizing the KinectFusion algorithm in real-world applications. Industries that can not afford the large and expensive three-dimensional modeling systems may find this technology suitable for their specific use cases.

Additionally, this work represents a start to the research that will be performed on this algorithm. While the experiments presented here are limited to the Kinect sensor and a scale model, the KinectFusion algorithm may be applied to other sensors and applications. The results presented here demonstrate how the model created with a Kinect sensor was improved ten-fold. What then might be possible if the KinectFusion algorithm were applied to higher resolution imaging systems? Perhaps then such systems might easily characterize surfaces with more precision and accuracy then is possible with the unaided human eye.

II. Background

This chapter provides the necessary background information for topics in three-dimensional modeling, sensor technologies, and modeling algorithms. Section 2.1 discusses of the basic building blocks of three-dimensional models, including the principles of point clouds, meshes, and other related topics. Section 2.2 provides a detailed overview of the two types of three-dimensional sensor technologies that are pertinent to this thesis: structured light ranging and coordinate measuring machines. Finally, the requisite algorithms used for building and comparing models are reviewed in Section 2.3.

2.1 Models

The field of three-dimensional modeling contains numerous terms and concepts that must be adequately defined before discussing the details of the sensors and algorithms used to build and manipulate the three-dimensional models seen in this work.

2.1.1 Point Clouds.

The term *point cloud* describes a large collection of points in multi-dimensional space. In a point cloud, there is no information regarding the connectedness of points, nor is there any other information that indicates shape or surface. Figure 2.1a provides an example of a point cloud based on the Stanford Bunny. The Stanford Bunny model is widely used for testing the capabilities of three-dimensional computer graphics systems and algorithms. Interestingly, the original bunny (Figure 2.1d) was a Terra cotta clay lawn ornament originally introduced by Turk and Levoy [28] for a project on creating polygon meshes from multiple range images. Since then, the bunny model, along with the Utah teapot [9], have become the canonical subjects for countless graphics tests, three-dimensional printings, and other three-dimensional research projects.

2.1.2 Meshes.

Although point clouds do not contain concepts of connectedness or surfaces, various algorithms exist for building *polygonal meshes* from the point clouds. The term polygonal meshes indicate that the points within a point cloud have now been associated with other points of the cloud to define polygonal shapes. The most common polygon used in three-dimensional modeling is the triangle, which will be used throughout this work.

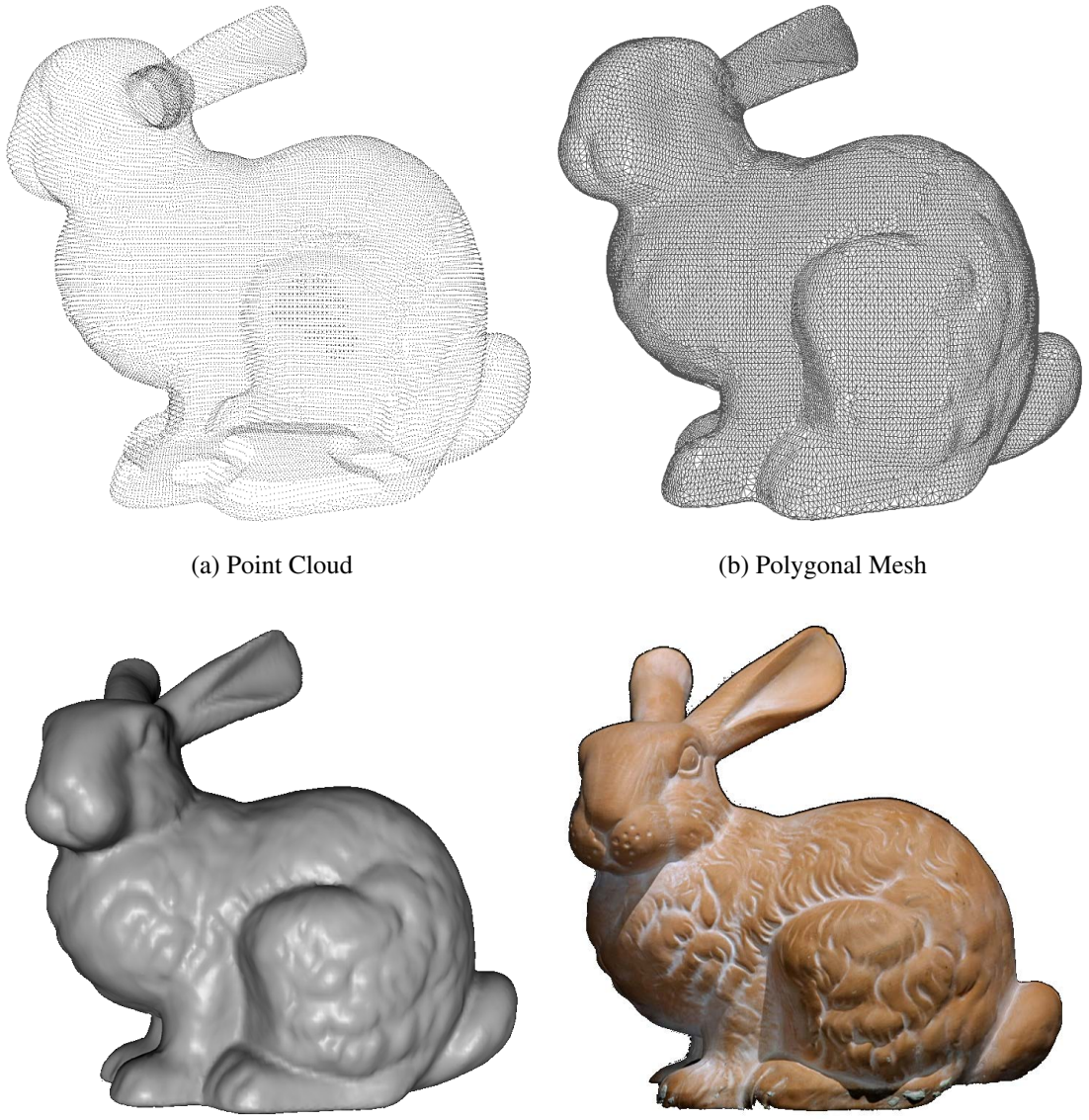
With the association of the points complete, edges (lines) are then defined between points of the same polygon. With edges defined, the points that make up the polygon are no longer referred to as points but as vertices, which signify that multiple edges meet at that location. This process is continued for all the points of a point cloud, and vertices now belong to multiple neighboring polygons. With vertices and edges defined, the face of the polygon, or the area bounded by the vertices and edges, can also be defined. All three pieces (vertices, edges, and faces) combine to represent the surface of the scanned object. An example of a polygonal mesh (in this case, a triangular mesh) of the Stanford Bunny is seen in Figure 2.1b.

2.1.3 Additional Terms.

After constructing a mesh, additional algorithms are applied to fill in gaps, close holes, and smooth the transitions from one polygon to another. The resulting object is no longer referred to as a mesh, but as a model. In many cases, the model only describes the outer surface of the object scanned (the model is hollow), and is called a surface model. A smoothed surface model of the Stanford Bunny is shown in Figure 2.1c.

2.2 Sensor Types

Numerous different types of three-dimensional sensors exist, each based on different technologies. Of particular importance to this research is the structured light technology employed by the Microsoft Kinect family of sensors. The second technology, coordinate



(a) Point Cloud

(b) Polygonal Mesh

(c) Model

(d) Original

Figure 2.1: Stages of Three-Dimensional Modeling. In this work, the sensors first produce a three-dimensional point cloud (a). From the point cloud, a polygonal mesh can be defined, as seen in (b). A smoothed model with shading is then created from the mesh (c). Finally, a picture of the original Stanford Bunny (d) object that was modeled. Images of Stanford Bunny model, (c and d), were sourced from the Stanford Computer Graphics Laboratory [4].

measuring machines, provides precision measurements that will serve as the *ground truth* measurements for this research. The following sections discuss how each technology works and highlight some of their respective strengths and weaknesses.

2.2.1 Structured Light.

The field of three-dimensional imaging is vast and full of many different techniques and methods. In the early 1970's, many different researchers began utilizing a projected light source and a light receiver to generate range (depth) maps of objects and scenes of objects using the *triangulation method* [26]. The triangulation method works by having a static angle between the light projection source and the light receiver and then comparing the received light to the light that was emitted. The distance is calculated by finding the intersection of the projected light and field of view of the camera (Figure 2.2, sourced from [26]). For example, Shirai [27] introduced what was termed a vertical slit projector that projected a vertical beam of light onto an object. A video camera then captured the reflected light and finally calculated the distance to each point within the vertical beam of light. Such efforts marked the beginning applications of projected light for three-dimensional imaging.

By 1988, when Besl [7] performed a survey on the numerous different types of range imaging sensors available at the time, the use of projected light and triangulation had grown immensely. From 1988 to today, that trend has only continued, as the number of different methods of using projected light has exploded. There are now several sub categories within the projected light classification that utilize colored light, gray scale light, color coded stripes of light, and even random textures. These new areas now form the basis for projected structured light as a means of three-dimensional range imaging.

One of the most prevalent and inexpensive sensors that employs the structured light technique, and the one used for this experiment, is the PrimeSense technology [13] found in the Asus Xtion series of sensors and popularized by the Kinect camera for the Xbox 360 gaming console. The Kinect was originally designed and developed to be a gaming console

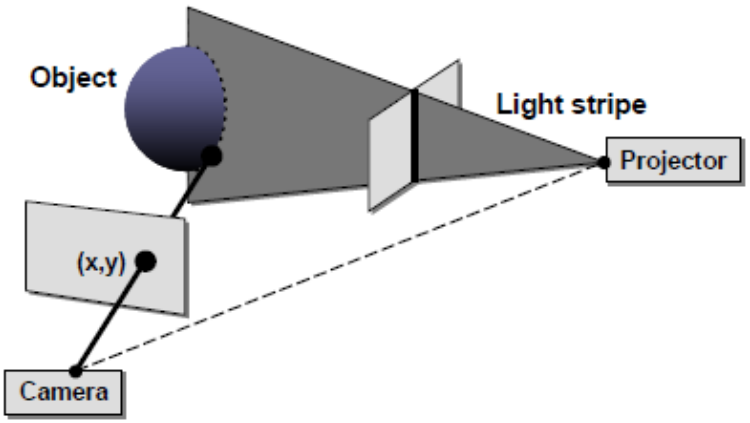


Figure 2.2: Projected Light and Triangulation Method for Ranging. This diagram depicts the method of projecting light from a source onto an object and viewing the projected light through a camera. To calculate the distance between the projector and object, the intersection between the camera's field of view and the plane of light created by the projector. Diagram sourced from [26].

accessory, adding the capability to interact with the console and games through speech and full-bodied movement in front of the sensor. To achieve such results, the Kinect consists of microphones to capture spoken commands, a 3-axis accelerometer used for determining the orientation of the Kinect, a standard Red, Green, Blue (RGB) camera for capturing the two-dimensional scene, and a system for capturing and processing the depth information of the scene. A diagram of the subsystems in the Kinect is seen in Figure 2.3.

The depth sensing system technology utilizes the triangulation and projected textures method to calculate depth images of a scene. According to several of the PrimeSense patents and publications [5, 14], the technology uses a light source to shine through a transparency, which then projects the texture pattern on the transparency out on to any objects in the scene. The reflected light of the scene is then received and focused by a lens in the Kinect onto the capture system, usually a Charge-Coupled Device (CCD) or Complementary metal oxide semiconductor (CMOS) based image sensor array. The

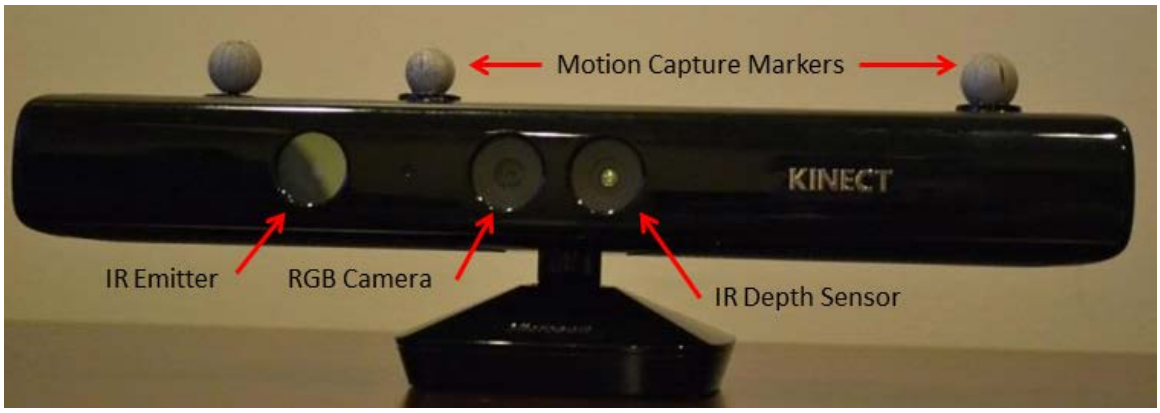


Figure 2.3: Microsoft Kinect System Components. The IR emitter shines through a transparency and projects the speckle patter of the transparency out into the field of view. The IR depth sensor receives the reflected IR light and then calculates the depth based on changes in the IR speckle pattern. The motion capture markers are not part of the standard Kinect configuration, and were added for this experiment.

Kinect's depth system utilizes an IR emitter and an IR receiver for the light source source and collector, which allows the system to work in varying lighting conditions. The projected reference pattern (Figure 2.4) is disturbed by the objects in the field of view and ultimately reflected back to the IR receiver, which receives the reflected IR beams, and the system then calculates the depth of the object based how the reference pattern is disturbed by objects in the scene.

The Kinect's IR depth system operates at 30 frames per second, where each depth frame is Video Graphics Array (VGA) resolution (640 horizontal pixels by 480 vertical pixels). Each pixel contains the Cartesian distance, in meters, between the sensor and the sensed object. This system can detect objects as close as 0.8 meters and as far away as 4.0 meters.

Khoshelham [19] found that the Kinect by itself has an accuracy of "few millimeters at 0.5 m distance to about 4 cm at the maximum range of the sensor [5.0 meters]". Also noted by Khoshelham and Elberink [20], is that while the mean and median distances between

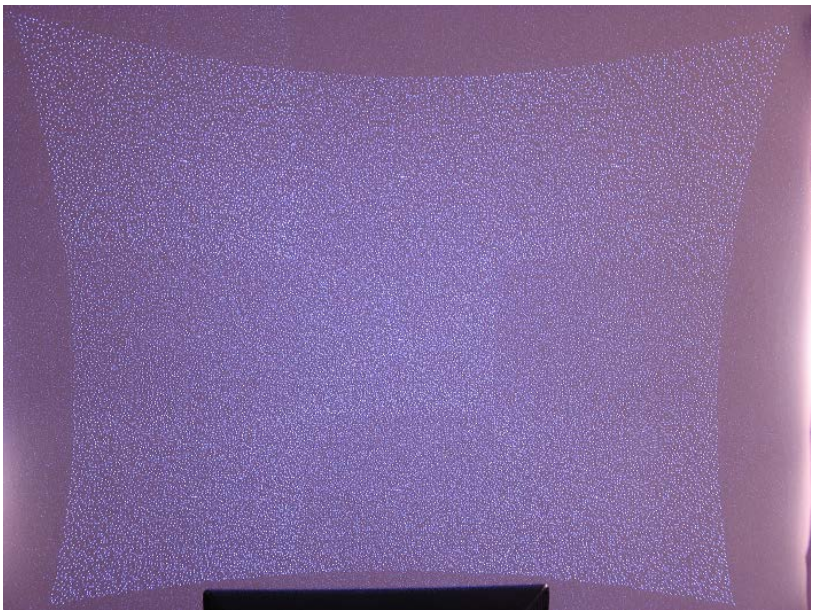


Figure 2.4: Kinect Speckle Pattern. This pattern is projected by the IR emitter of the Kinect. The reflected pattern is then received by the IR receiver, and compared to the known reference pattern that was projected. Based on the differences between the reflected pattern and the reference pattern, calculations are performed to determine the depth of objects within the scene. Sourced from the FuturePicture series on Kinect Hacking [22].

registered models are 0.1 and 0.0 cm respectively, the standard deviation for the depth values is between 1.0 cm, 1.1 cm, and 1.8 cm in the X, Y, Z directions, respectively.

After the release of the Kinect, many researchers began applying the Kinect to applications unrelated to gaming. Due to this popularity, Microsoft released a standalone sensor/camera, a variant of the Kinect for Windows-based development. This version, named *Kinect for Windows*, is identical in physical construction and system components, but differs in that the system firmware has been modified to support a closer operating range. This *near range* mode provides an operating range of 0.4 meters to 3.0 meters. Figure 2.5 presents a comparison of the operational distances of the Xbox 360 and Windows variants.

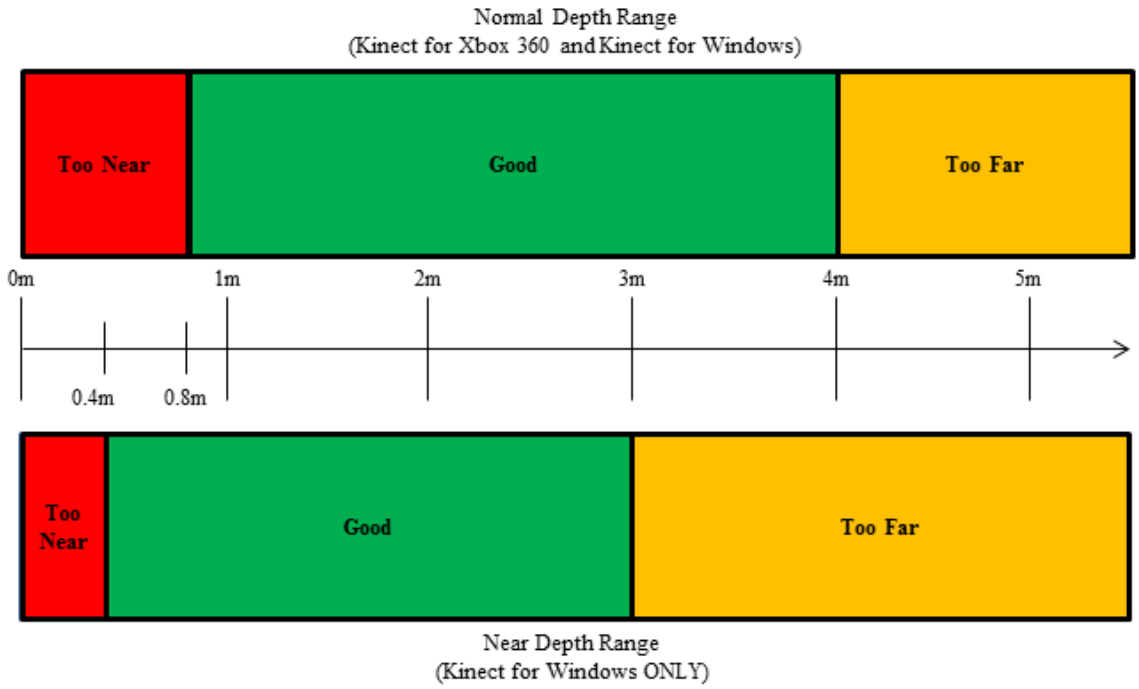


Figure 2.5: Kinect Depth Ranges. This diagram depicts the differences in the two range modes available in the Kinect for Windows variant of the sensor. Note that the Kinect for Windows can operate in either the normal or near range, and the original Kinect for Xbox 360 can only operate in the normal range.

Along with the release of the Windows variant, Microsoft also released a Software Development Kit (SDK) providing developers with access to the capabilities the Kinect sensor provides (i.e. color camera data stream, depth camera data stream, human skeleton tracking, audio capture, etc). The SDK also includes documentation and example applications that showcase the Kinect's capabilities. One of which is the KinectFusion algorithm that will be used in this experiment and is discussed in further detail later. Additionally, the Kinect for Windows SDK contains an application called *Kinect Studio* which can be used to record the data streams from the Kinect sensor and playback the recordings at a later time. The Kinect Studio application will be utilized to record the scanning sessions for this work, and play them back for model building.

2.2.2 Coordinate Measuring Machines.

A CMM varies from a structured light based sensor in that it does not utilize a projected light pattern and the resulting disturbances to calculate the depth measurement. In contrast, a CMM utilizes a probe at the end of at least three different axes of movement and measures the digression in each axis in order to calculate the location of the probe in three-dimensional space. Depending on the type and technical specifications of the specific CMM, the measurements will be orders of magnitude more accurate and precise than the Kinect sensor can be.

CMM systems scan an object by moving the probe around the object's surface, constantly keeping the probe in contact the object's surface. At each probe location, a range measurement is taken, which is then converted into a measurement consisting of the Cartesian X, Y, Z values relative to the CMM's point of origin. This process is continued over the entire surface of the object being scanned, gradually building a collection of three dimensional points. The collection of three-dimensional points forms a point cloud, and provides a rough estimate of the outer surface of the object.

For this work, a FaroArm Edge [12] CMM will be used to scan the subject model, providing a ground truth model from which to make comparisons against those models constructed with the Kinect sensor. The FaroArm series of tools are portable articulated arm type CMM devices, which consist of a base, multiple arms/links, multiple joints to create the different axes of movement, and a probe for touching or scanning the surface of the object. Figure 2.6 shows the basic concepts of an articulated arm type of CMM device. In the FaroArm line of devices there are six or seven joints, in which rotary encoders measure the rotational angle of each joint. These measurements, along with the fixed length of each section of the articulated arm, are combined to calculate the location of the probe's tip in three-dimensional space.

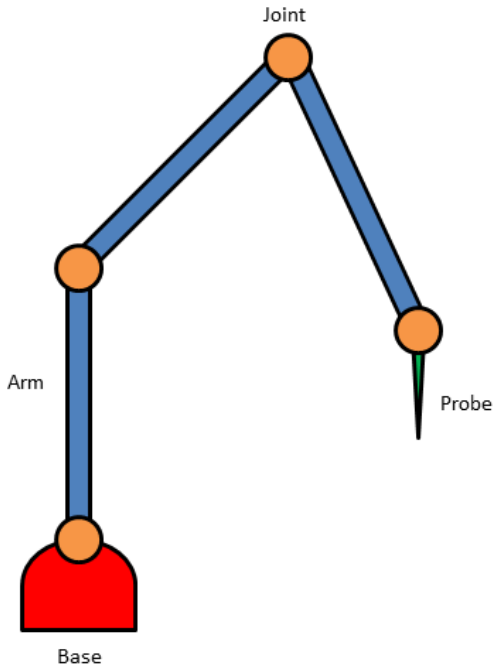


Figure 2.6: Simple Articulated Arm Coordinate Measuring Machine. A basic articulated arm CMM consists of a base, numerous joints and arms, and a probe. The joints contain encoders for measuring the amount of rotation in the joint, and the arms have known fixed lengths. These two pieces of information are combined in order to calculate a three dimension coordinate for the tip of the probe.

One key limitation of the FaroArm is that the base of the FaroArm Edge must be stationary, as the calculations for the three-dimensional coordinates are based on the movement relative to the base. Secondly, given the stationary requirement of the base, articulated arm based CMM's have an inherent limitation on the size of the object that can be scanned. With fixed length arms, the volume of space that can be reached by the probe is also fixed. In the case of the FaroArm series, the volume of space available for scanning ranges between 1.8 and 3.7 cubic meters. For this experiment, the particular FaroArm Edge utilized has a volumetric reach of 2.7 cubic meters. While this volume is sufficient for this experiment work with a scale model, it is not sufficient for a full size aircraft.

2.3 Algorithms

Three-dimensional sensors like the FaroArm Edge and Kinect provide the means of obtaining range measurements or calculating Cartesian coordinates, but do not directly create three-dimensional models. Specialized algorithms are employed either in hardware or software to combine the measurements or points from the sensors into polygonal meshes or models that represent the surface of the sensed object. Along with the modeling algorithms, numerous other algorithms exist that are used for building, cleaning, manipulating, and comparing three-dimensional models, which should be reviewed.

2.3.1 Iterative Closest Point Algorithm.

While not the focus of this work, the ICP algorithm plays a role in various parts of this work. In general, the ICP algorithm provides a method for registering (aligning) three-dimensional shapes, including free-form curves and surfaces. Besl and McKay [8] originally proposed the ICP algorithm for the registration of a given *model* shape and a sensed *data* shape. Although numerous variations of the original ICP algorithm have been introduced since the initial publication of ICP [25], this work uses the original algorithm.

The ICP [8] process consists of four steps: (1) compute the closest points, (2) compute the least squares registration between closest points, (3) apply registration, and (4) terminate iteration when the metric falls below a preset threshold. The first step consists of calculating the closest points between two point sets, one representing the data shape and the other representing the model shape. For each point p in the data shape, the distance between p and every point in the model shape is calculated by computing the Euclidean distance and taking the minimum value. The second step estimates possible transformation functions, consisting of both translation and rotation, that yield the smallest mean square difference between the data points and model points. The third step applies the transformation with the least mean square difference to the data shape's points. Finally,

the fourth step starts the process over unless the mean square difference is below a preset threshold.

ICP is applied in later experiments to align models constructed using KinectFusion against a model constructed by the FaroArm Edge CMM. Furthermore, ICP is also used within the KinectFusion algorithm to estimate the sensor's position by comparing the current image against the previously constructed dense surface model.

2.3.2 KinectFusion.

In work presented by Izadi et al. [17], the KinectFusion algorithm demonstrated the ability to create three-dimensional models of objects and scenes from the Kinect sensor. The algorithm creates a full mesh model of the object by combining the sequential depth images of an object from varying viewpoints.

Although three-dimensional modeling and reconstruction is not a new topic, the KinectFusion approach combined several novel goals that had never previously been combined before. The goals of the KinectFusion algorithm were to utilize interactive rates, have no reliance on feature detection, provide a high quality reconstruction, allow for dynamic interaction, have no infrastructure requirement, and be able to operate on a room size scale [17].

Previous three-dimensional modeling systems relied on *off line* reconstructions where the sensor captured the information and then processed the information into the full three-dimensional model at a later time. With improvements in Central Processing Unit (CPU) and Graphics Processing Unit (GPU) capabilities, such models can be built in real-time, but are not fast enough to incorporate user interaction with the model. This goal is extremely important to any visual inspection type usage, as with current practices employed, inspectors are able to interact with the object and the results of their inspection in real-time. For instance, if an inspector finds that they need additional details or clarification on a certain portion of the object, he/she simply revisits the area while performing the

inspection. Systems that consist of non-interactive modeling rates would be less helpful than the current off line practices.

Similar systems in use today rely on detecting a specific feature or set of features in the scene that is being scanned in order to help build the full model. The KinectFusion algorithm is able to remove this requirement by utilizing pose estimation (discussed in the next section) techniques that enable the system to track the sensor location and it's perspective relative to the scene. The KinectFusion algorithm has proven itself to be attractive based on its ability to pick up a sensor and start scanning immediately (i.e. without any prior knowledge or information about the object or scene), which opens the door for the usage of the KinectFusion algorithm to a host of new applications. This is a helpful feature for the visual inspection scenario wherein the characteristics of the defects being searched for may no be known a priori.

The goal from Izadi et al. [17] that will be examined by this work is the ability to build high quality reconstructions. As this is a goal for all systems in three-dimensional modeling, it requires a quantification of the quality that can be achieved with the system. That is what this work will provide.

Additionally, Izadi et al. [17] wanted the algorithm to handle dynamic, or changing, scenes. Previous systems required that the scene or object being scanned be static. However, in the visual inspection case, the object will likely be a large stationary scene, and so the dynamic scene goal is less important. The dynamic nature of the KinectFusion is also closely related to the goal of being infrastructure-less. KinectFusion does not need a large setup, multiple sensors, or a long logistical tail in order to be used. This is one of the most important requirements for any tool in today's expeditionary force. In the USAF, inspections occur both at home bases and deployed locations, thus requiring such systems be able to deploy and redeploy quickly. Rather than setting up a system of multiple cameras and performing fine tuned calibrations, a system consisting of a single hand-held mobile

sensor and a computer running KinectFusion is a better option to augment the requisite visual inspections.

The final stated goal for the KinectFusion algorithm [17] was the ability to operate on a room scale. Specifically, the goal was for a user to be able to pick up a sensor and walk around the interior of a room while creating a reconstruction of the entire scene. Previous systems and algorithms were limited to much smaller scales, such as desk size. A system based on the KinectFusion algorithm's room scale capabilities now allow it to be used on both the small scale items and room scale and larger items such as an entire aircraft [10].

As the stated goals for the KinectFusion algorithm align with the motivation to provide three-dimensional sensor-based visual surface inspections, an in depth analysis of the methodology and the capabilities of the algorithm is merited. To that end, Izadi et al. [17] describe the KinectFusion algorithm as four main stages: (1) depth map conversion, (2) camera tracking, (3) volumetric integration, and (4) raycasting (Figure 2.7).

The depth map conversion stage works on the image coordinates from the current depth map (Figure 2.8), which is a single VGA resolution (640x480 pixel) frame of range data. At each pixel location (X,Y) in the depth map image, there is a range value. Each of the image coordinates and their respective range values is then converted to a three-dimensional point (X, Y, Z) relative to the Kinect's coordinate system. Along with the three-dimensional point, a normal vector is calculated for each point.

The second stage, camera tracking, relies on the ICP algorithm (Section 2.3.1) to align the three-dimensional points computed in stage-one with the converted points from the previous iteration of the algorithm. This allows the three-dimensional points to be converted to a single global coordinate system, which allows the third stage to progress.

In the volumetric integration stage the new three-dimensional coordinates are applied to the volumetric representation of the object being scanned. The representation consists of many voxels, which are similar to pixels in two-dimensional space, but are represented

as rectangular prisms in three-dimensional space. Each voxel contains the average of the depth range values from all the points that have been mapped to that voxel by the camera tracking and depth map conversion steps.

Finally, the updated volumetric representation is raycast, i.e. rendered, to generate the necessary views of the current state of the model. The rendered version of the model is then fed back into the volumetric integration step for averaging with the set of three-dimensional points that will arrive in the next iteration. Additionally, the rendered model can be fed into the camera tracking stage, so that the ICP algorithm can align the next set of points with the higher fidelity model, as opposed to just a aligning with a another set of points. This four step process is repeated for every new depth map received from the Kinect sensor.

2.3.3 Pose Estimation.

Pose estimation is the term used to describe several different algorithms that determine the location of a camera based on a picture or frame of video. Pose estimation is vital to the KinectFusion algorithm's three-dimensional modeling process. In order to fuse the points from two perspectives, it is important to determine the difference in the two perspectives. If there is no difference, then the two sets of points are simply averaged. This is not the case if the perspectives are different, as the sensed points of the second perspective are not the same points from the first perspective, and should not be averaged. Overall, correctly estimating the pose of the sensor allows the depth map coordinates to be converted to a global coordinate system, which then forms the three-dimensional model [17].

The KinectFusion algorithm relies heavily on the ICP algorithm for help in tracking the sensor's location. KinectFusion starts with an existing estimate of the sensor's position (i.e. from the previous frame or the initial setup position). Each point in the previous frame's vertex map is transformed into camera coordinate space, and then projected into image coordinates. These two-dimensional points are then used to lookup corresponding points in the current frame's vertex map. Finally, for each pair of corresponding points,

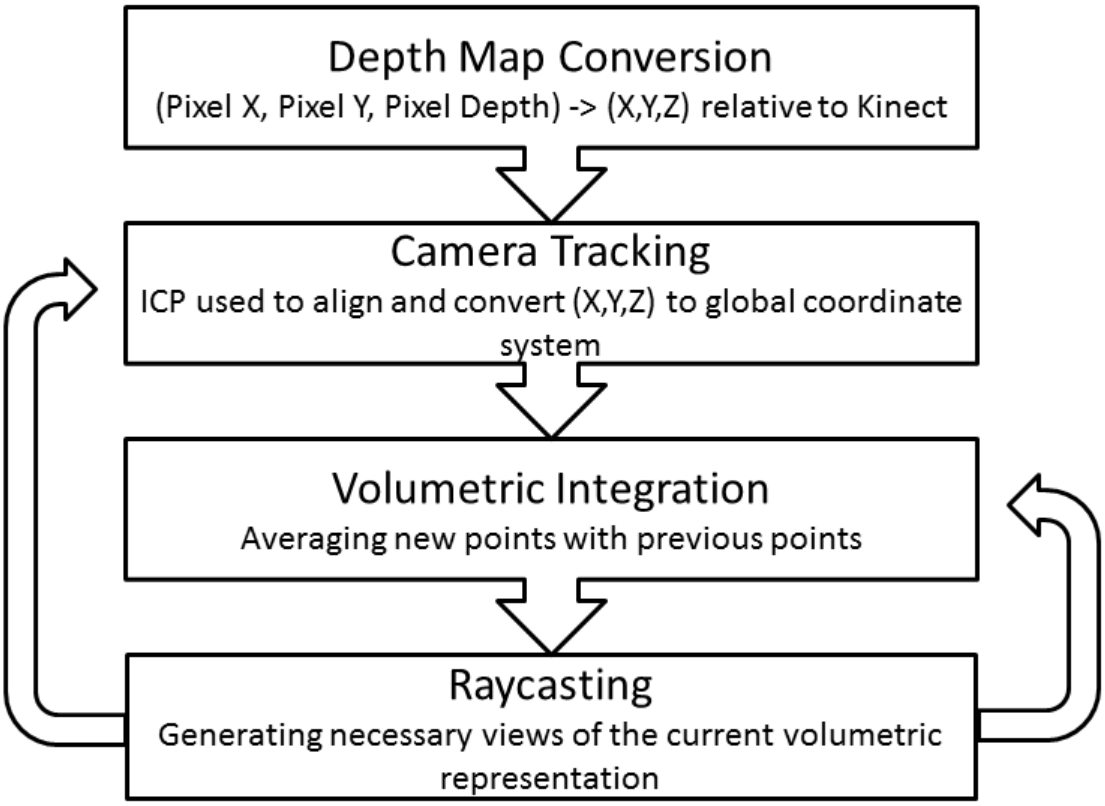


Figure 2.7: Stages of KinectFusion Algorithm. The four steps of the KinectFusion algorithm. The pixel coordinates and depth ranges are converted to three-dimensional points relative to the Kinect sensor. The points are then aligned with the previous frame's points or the current model and then converted to a global coordinate system. Once in the same coordinate system, the points are averaged into the current model in the volumetric integration stage. Finally, the curret status of the model is raycast, or rendered, for viewing and fed back to the camera tracking stage of improved alignment and to the integration for stage for the next round of integration.

the algorithm rejects any outliers by testing their Euclidean distances and that the angles between points are within a predetermined threshold. This process is then repeated for each new depth frame [17].

In version 1.8 of the Kinect for Windows SDK, there are two distinct methods of calculating the pose of the Kinect sensor. The SDK documentation [2] refers to them as AlignDepthFloatToReconstruction and AlignPointClouds. Based on the descriptive

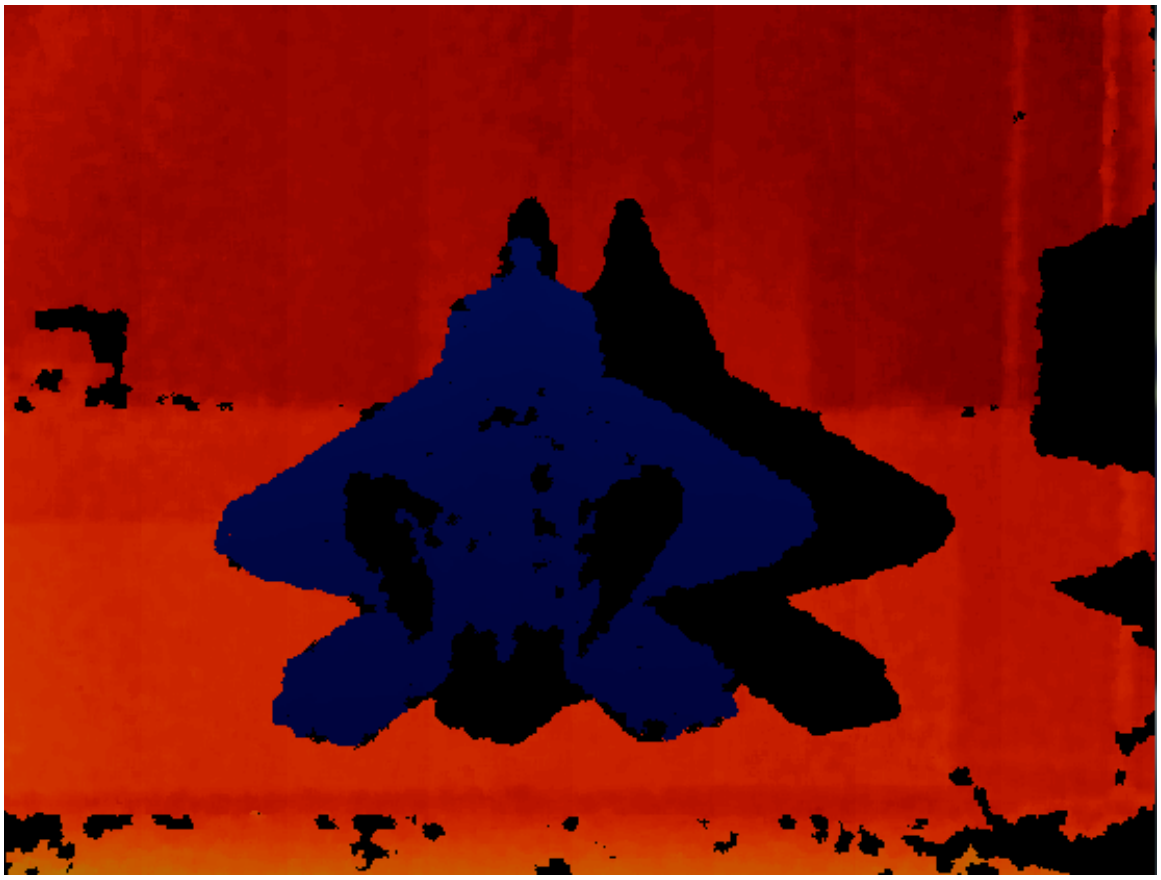


Figure 2.8: Kinect Depth Map. The depth map produced by the Kinect sensor. Blue colors indicate close ranges (0.4-0.6 meters), red indicate further away range values (3.0-4.0 meters), and black represents a range value of 0, which indicates an invalid range measurement.

names, AlignDepthFloatToReconstruction compares the current depth image (collection of range measurements in a single frame) to the currently built three-dimensional model (or reconstruction). On the other hand, AlignPointClouds does not require an existing reconstruction model, and will simply compare one point cloud to another.

For creating a surface reconstruction with the KinectFusion algorithm, it is recommended to use the AlignDepthFloatToReconstruction function, as it provides more accurate camera tracking results in this scenario. This is recommended because the fused model will have higher accuracy information than a single point cloud. However, the function has

trouble with objects moving in the reconstruction scene, so the AlignPointClouds function is recommended for scenes with mobile objects.

2.3.4 Hausdorff Distance.

While ICP is vital to finding the best possible alignment between two similar models, it provides little if any quantification for how similar the resulting registered models are to each other. Thus a metric capable of quantifying how closely the two models are to each other is required.

With the alignment of two models set using the ICP algorithm, it becomes possible to calculate the Euclidean distance between any pair of registered points on the models. By performing this calculation on a large sampling of points or even the entire model, it possible to quantify how similar two models are to each other. This process utilizes the Hausdorff distance or metric, and has been applied to simple set comparisons [15, 23], two-dimensional image comparisons [16], and now most recently, three-dimensional model comparisons [6, 11].

In this work, the difference between two models will be measured using the Hausdorff distance (originally presented by Hausdorff [15]), which is a metric for quantifying the distance between two subsets of a metric space. The Hausdorff distance describes the maximum distance of a set to the nearest point in a second set. Consider two sets, $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$. The Hausdorff distance between A and B is denoted as

$$H(A, B) = \max\left\{\max_{a \in A} \min_{b \in B} |b - a|, \max_{b \in B} \min_{a \in A} |a - b|\right\} \tag{2.1}$$

such that the metric finds the smallest distance d where every point in A has a point in B within distance d and every point in B has a point in A within distance d [23]. Huttenlocher et al. [16] demonstrated how to use the metric for comparing two-dimensional images, which then inspired Cignoni et al. [11] and Aspert et al. [6] to apply the Hausdorff distance to three-dimensional models.

III. Methodology

A visual post flight inspection of the surface of low observable aircraft looks for deviations from the standard configuration. Such tedious tasks require multiple personnel to comb the aircraft surface looking for minute defects. Despite the tedious nature of the inspections, they play a large part in determining future maintenance work load and the potential performance of the aircraft. It is the hope of this work to inspire automation that will relieve this burden.

Three-dimensional sensors and modeling algorithms provide a glimmer of hope that the labor requirements for such inspections could one day be reduced. However, before any human eyeball based visual inspections can be replaced with computer vision based inspections, the accuracy and precision of such sensors and algorithms must be clearly determined. With a clear understanding of the accuracy and quality of the available sensors and algorithms, the appropriate devices and methods can be selected for the right inspection tasks.

This work has two main goals. The first goal aims to quantify the accuracy of the three-dimensional modeling capabilities of the KinectFusion algorithm when paired with the Kinect sensor. Previous work and experiments have shown qualitative model improvements by utilizing the KinectFusion, but no work has yet quantified the level of improvement. This work will measure the level of improvement provided by the KinectFusion algorithm by comparing models constructing using the algorithm against models constructed with a high-accuracy CMM device.

The second goal of this research is to determine if the KinectFusion algorithm, in conjunction with the Kinect sensor, can detect items smaller than two millimeters, and if so, how well are those objects modeled. To evaluate this goal, models will be created of a static scene of United States (US) coins in stacks of varying thickness. The models

will then be analyzed to determine what thicknesses were modeled, and thus allowing the modeled thickness to be compared with the thickness of the physical object being scanned.

This chapter provides the details of the two experiments performed here: the *model accuracy experiment*, and the *anomaly detection experiment*. Each experiment will be presented in detail in its own section, with subsections discussing the general experiment setup, preliminary testing and results, experimental design, metrics, parameters, evaluation technique, and process.

3.1 Model Accuracy Experiment

The goal of the model accuracy experiment is to quantify the level of improvement that is achieved by the KinectFusion algorithm when compared to the Kinect by itself. The accuracy of the Kinect sensor was highlighted by Khoshelham [19], while the quality of the KinectFusion based models will be determined through this experiment.

3.1.1 Experiment Setup.

To quantify the quality and accuracy of a KinectFusion based model, there must be comparison of a KinectFusion model to a reference model with very high accuracy. The subject for this work, a 1/48th scale model of a US F-22 airplane, will be scanned and modeled by the FaroArm Edge CMM device. The FaroArm Edge device yields an accuracy of ±0.041 mm [12], while the Kinect itself yields an accuracy of approximately ±10 mm [20]. Given the significant difference between the FaroArm Edge and Kinect, this experiment will consider the FaroArm Edge based model to be the reference model.

The subject will then be scanned by the Kinect sensor and the KinectFusion algorithm will be utilized to create the three-dimensional model of the subject. To build complete three-dimensional models, the sensor will be rotated around the static subject model. This rotation provides the various sensor perspectives needed to build a complete model. The KinectFusion models will be created for varying distances between the sensor and subject, and also at varying resolution settings at each distance. Exploration of these two parameters

is expected to highlight how each affects the overall quality of the three-dimensional model that can be built using the KinectFusion algorithm. For this experiment, a Kinect sensor and a computer with powerful GPU is required. The computer will provide the platform for utilizing the KinectFusion algorithm, which has been implemented in several of the sample applications provided in version 1.8 of the Kinect for Windows SDK. In particular, the sample called *KinectFusion Explorer-WPF C# Sample* provides the ability to generate and export three-dimensional models while also adjusting the parameters of the KinectFusion algorithm to achieve different results.

To provide the rotation and varying sensor distances, a specialized setup was created from a motorized turn table. From there, a sensor mount was created from two bars of aluminum, one mounted to the horizontal surface of the turn table and running through the center of the rotational motion, the other mounted perpendicularly at the end of the first bar. The Kinect was then mounted to a Kinect-specific tripod mount, which was mounted to the vertical bar. The horizontal bar can be shifted to vary the operating distance of the sensor. The tripod mount of the sensor can also be adjusted to provide different viewing angles if necessary. The speed and direction of the turn table's rotation is controlled using a Universal Serial Bus (USB) based stepper motor controller and the corresponding software. A picture of the turn table and sensor mount can be seen in Figure 3.1.

With the models built, they will be aligned/registered, and then compared using the calculated Hausdorff distance metric for every vertex in the models. The calculated distance metrics will also be used to generate *heat maps* of each KinectFusion model, which will highlight the varying distances across the entire surface of the subject. Finally, the statistical analysis of the Hausdorff distances and the visual inspection of the resulting heat maps will provide the basis for determining the level of improvement achieved by the KinectFusion algorithm.

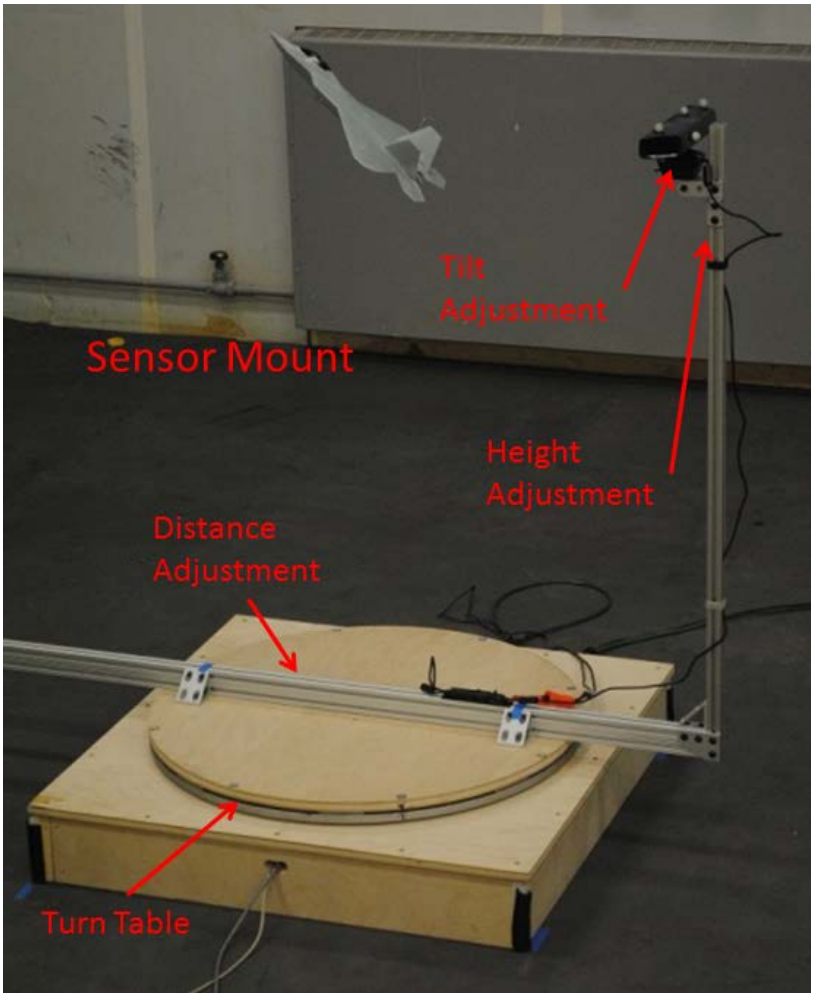


Figure 3.1: Turn Table and Sensor Mount.The turn table and custom sensor mount used to provide full 360° rotations about the suspended model. The horizontal bar is adjusted to change the sensor distance, while the height and tilt adjustments were not utilized in this experiment.

Additionally, the entire experiment setup will be built and operated in a specialized motion capture lab. The Vicon motion capture system utilizes an array of high speed IR cameras to capture the location and track the motion of special IR reflective markers. Overall, the Vicon system provides for sub-millimeter accuracy. The Kinect will be modified to add four IR reflective markers for tracking, and the entire experiment will be conducted in the Vicon lab. Each Kinect scan session will also be recorded by the

Vicon system, which will capture the precise location of the Kinect sensor as it moves around the static subject. This data can then be used later to investigate the accuracy of the KinectFusion pose estimation techniques (Section 2.3.3).

3.1.2 Preliminary Testing, Results, and Findings.

Preliminary tests were conducted to determine several experiment parameters that would then be fixed for the final experiment runs. Parameters to be determined included sensor distances, GPU required for real-time KinectFusion usage, the rotational speed of the turn table, and parameters of the KinectFusion algorithm.

3.1.2.1 Sensor Findings.

In order to determine the possible sensor distances to later be used as factors in the experiment, several tests of the experiment setup were run with the Kinect for Xbox 360 sensor, the only sensor on-hand at the start of the testing phase. As discussed in Section 2.2, the Kinect for Xbox sensor has a distance range of 0.8 meters to 4.0 meters. Testing at 0.8 meters rarely finished with a full model being generated. In the vast majority of early tests, the system would reconstruct approximately half of the airplane before *losing track* of the physical model, a failure of the pose estimation method, which results in a failed reconstruction (though half of the plane was modeled correctly, the other half was not, and thus yielded a incomplete model that was insufficient for further comparisons).

The problem was suspected to be the size of the 1/48th scale model in relation to the operating distance of the sensor. The airplane occupies a *bounding box* in three dimensional space that is only 0.3938 meters long by 0.2825 meters wide by 0.1058 meters high. For simplicity, it is best to consider this as a solid rectangular box in three-dimensional space, when in all reality the entirety of that rectangular box is not actually occupied by the scale model. For example, the 0.3938 meter height is based on the height of the tail fins, while the rest of the airplane model is not 0.3938 meters high. To verify that the size of the model

compared to the operating distance was indeed the problem, calculations were performed to determine how much of the Kinect's field of view was occupied by the model airplane.

The Kinect sensor has a 43° vertical field of view and a 57° horizontal field of view, both centered at the IR depth sensor described in Section 2.2. Based on the sensor information, the front plane based on a 0.8 meter distance between the Kinect sensor and the airplane, is 0.6303 meters high by 0.8687 meters wide, with a surface area of 0.5475 meters squared. With it's broadest face presented to the sensor (i.e. the airplane nose is pointed straight up and the wings spanning side to side), the 1/48th scale model only presents a surface area of 0.1112 meters squared (0.3938 meters by 0.2825 meters). Therefore, at the best possible viewing scenario with the sensor at 0.8 meters away from the aircraft model, the airplane only occupies 20.3159% of the Kinect's sensing plane. Given the above is based on the idea of a solid rectangular bounding box, the percentage occupied by the plane is actually less than 20.3159%. In the worst case, when the airplane's nose is pointed away from sensor and the tail section is viewed from directly behind, the airplane only occupies 5.4604% of the sensing plane (0.2825 meters by 0.1058 meters = 0.0299 meters squared surface area).

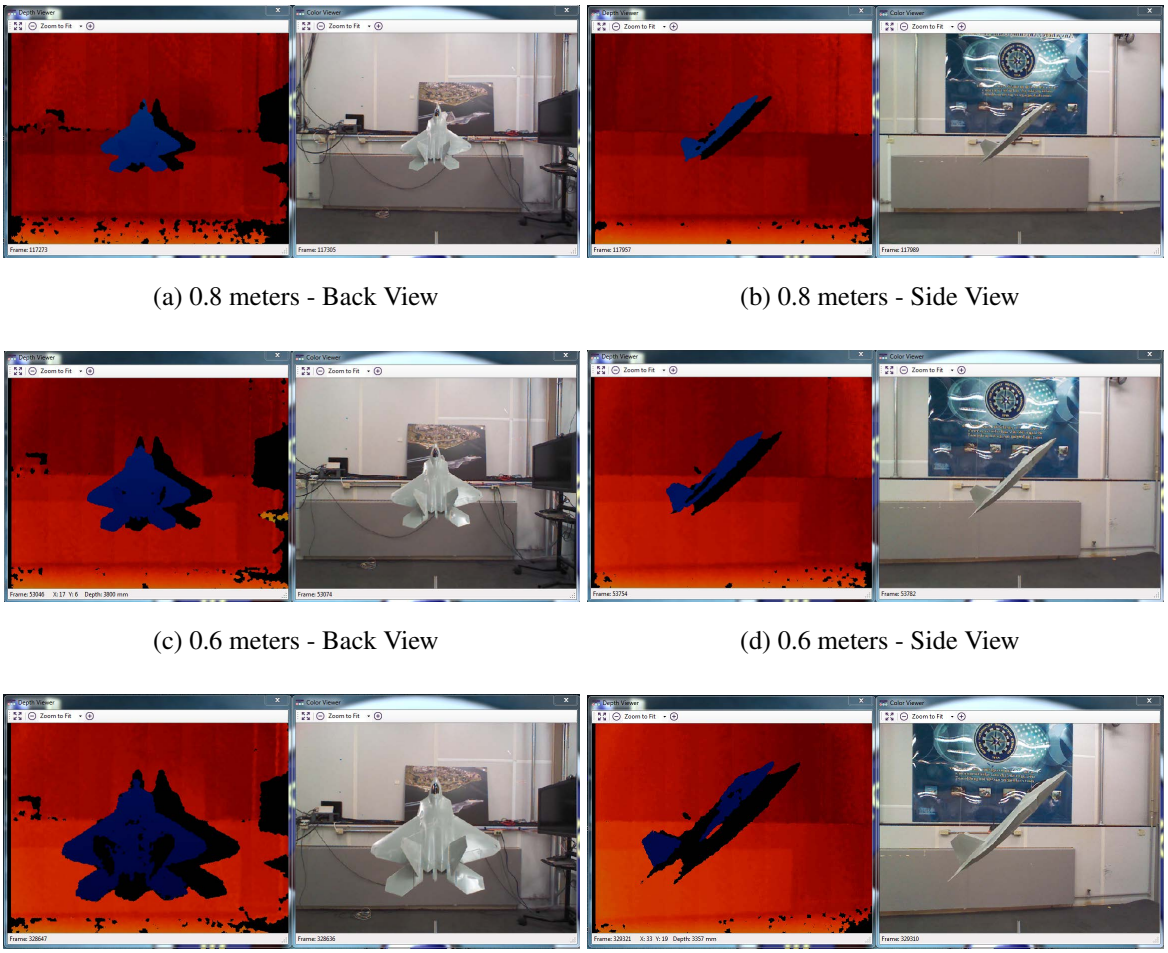
This highlights the cause of the pose estimation failures during the scanning process. As the sensor rotates around the airplane, there are two diametrically opposed points at which the viewing surface area of the plane is at an absolute minimum, and unfortunately at 0.8 meters, the airplane model's surface presented to the sensor is so small in comparison to the sensors field of view, that the pose estimation method can fail and the model build will fail. It is possible to stop and reverse the rotation of the sensor to align the sensor with the last successful perspective that was added to the model, at which point the model building can continue. However, this can only be accomplished at certain settings, after which no matter how many times the model is realigned with the physical subject, the

algorithm simply can not track the camera's pose any further. The settings at which this occurs will not be included in this body of work.

Due to the range restriction of the Kinect for Xbox 360 sensor, a new Kinect for Windows sensor was obtained, which provides the ability to operate in a near mode consisting of 0.4 meters to 3.0 meters. From the same calculations as used for the Kinect for Xbox 360 sensor, at 0.4 meters, the front sensing plane is 0.3151 meters high by 0.4344 meters wide, yielding a surface area of 0.1369 meters squared. As the surface area of the airplane model does not change, it's surface area from the best viewpoint is still 0.1112 meters squared, which is 81.2638% of the sensing plane at 0.4 meters. This is a dramatic, almost four-fold increase in the amount of surface area that will provide valid depth results. At the worst viewpoint, a surface area of 0.0299 meters squared, occupies 21.8416% of the sensing plane, a similar four-fold increase in sensing plane occupation. A comparison of the sensor's field of view at 0.8 meters, 0.6 meters, and 0.4 meters can be seen in Figure 3.2.

While a further, in depth analysis of model volume vs the Kinect's frustum volume could be calculated and analyzed, the above calculations based on the simplified surface area indicate a significant improvement by operating at the 0.4 meter distance, while the volumetric calculations would only confirm with greater accuracy the level of improvement. Due to this, the volumetric calculations were not performed.

Given the improvements with the closer operating range, testing resumed with a new Kinect for Windows sensor at the 0.4 meter range. The new sensor and operating range encountered very few issues, which yielded complete models suitable for comparison with the CMM generated model. Based on the new sensor's range, the calculations were computed for distances between 0.4 meters and 0.8 meters to determine the best options for generating models. At 0.6 meters, the model airplane occupies between 36.1172% and 9.7074% of the sensing plane, which presents a significant difference from 0.4 meters and an excellent distance for comparisons. Finally, 0.8 meters was chosen as the farthest



(a) 0.8 meters - Back View

(b) 0.8 meters - Side View

(c) 0.6 meters - Back View

(d) 0.6 meters - Side View

(e) 0.4 meters - Back View

(f) 0.4 meters - Side View

Figure 3.2: Subject View Size at Varied Distances. These images show how much of the sensor's view is taken up by the 1/48th scale model at the 0.8 meter (a and b), 0.6 meter (c and d), and 0.4 meter distances (e and f). The left image in each figure is the Kinect depth image, and the right is the Kinect RGB view.

distance factor, since the KinectFusion algorithm could still build some of the lower resolution models that would be built at the 0.4 meter and 0.6 meter distances.

Additionally, calculations were performed to highlight the differences that would be caused by a larger model. For instance, a 1/12th scale P-51 model would be too large to

fit in the sensor's field of view at 0.4 meters, and even so large that the closest distance possible would be approximately 1.0 meters.

3.1.2.2 GPU Requirement.

Based on the recommendations of the KinectFusion developers[2], an NVidia GeForce GTX560 or better card is necessary to perform real-time KinectFusion surface modeling. The GTX560 consists of 336 Compute Unified Device Architecture (CUDA) cores, which provide parallel processing pipelines to help distribute the workload of the KinectFusion algorithm's calculations. As the number of processing pipelines, or in the case of NVidia based graphics cards, CUDA cores increases, the more data the card can process per second which in turn provides an increase in the frames per second (fps) performance of the KinectFusion algorithm.

An NVidia Tesla K20 GPU was initially used for testing, as the K20 has 2496 CUDA cores and includes 5GB of GDDR5 memory. Based on the basic technical specifications, the Tesla K20 card should have provided excellent performance with the KinectFusion algorithm. Unfortunately, upon further inspection, the Tesla K20 card does not support the C++ Accelerated Massive Parallelism (AMP) library in DirectX 11 and therefore can not be utilized for the KinectFusion algorithm.

Older versions in the Tesla series of GPUs do support the C++ AMP library, so an older Tesla card was then obtained, the C2075, which offered 448 CUDA cores and 6GB of GDDR5 memory. Initial testing revealed that the C2075 GPU is capable of building models in real-time with KinectFusion, but the highest frame rate was only 15 fps, and quickly slowed down to between 3 and 8 fps. Unfortunately, during the slow down of the C2075, the pose estimation algorithm would lose track of the model, caused by the dropped frames, and the model build would fail.

The GeForce GTX Titan, one of the highest performing GPUs currently available, offers 2688 CUDA cores, 6GB of GDDR5 of memory and support for the C++ AMP. In

preliminary tests, the GTX Titan supported real-time model construction with frame rates in the upper 25-28 fps range, and successful model building runs.

Along with the parallel execution cores of the GPUs, the memory available on the card is extremely important for utilizing the KinectFusion algorithm. According to the Kinect for Windows SDK documentation [2], the algorithm will model a volume of three-dimensional space whose size and resolution is governed by the parameters selected in the sample application. The reconstruction volume is made up of rectangular prisms of varying size and resolution and are referred to as voxels (similar to pixels but in three-dimensional space). The number of voxels that can be constructed is governed by the amount of memory available to the algorithm. For running in real-time with the GPU, the video memory of the GPU will dictate the voxels, not the system memory. Secondly, the memory must be a single continuous block of memory, which is rarely the full advertised onboard memory size. From the SDK documentation, the typical upper bound on number of voxels is around $640 \times 640 \times 640$, or $262,144,000$ voxels, which requires approximately 1.5GB of memory. However, with the GeForce GTX Titan's 6GB of Random Access Memory (RAM), it was possible to get as high as $640 \times 448 \times 1600$, or $458,752,000$ voxels. This number used approximately 3433 MB of the Titan's RAM.

3.1.2.3 Sensor Rotational Speed.

Along with the sensor and GPU capabilities, it was also important to determine the optimal speed of rotation of the sensor. The motor controller software measures speed in pulses per second, not in number of revolutions per time period. The speed range is approximately 100 pulses to 2000 pulses. In the preliminary testing, the slower the rotational speed of the sensor, the better the model quality, likely due the increased time obtaining sensed depth values for each section of the airplane. The optimal speed found was 300 pulses per second, which was then used to time complete revolutions of the table, which yielded a speed of 1 revolution every 96 seconds, or 0.625 revolutions per minute (rpm).

3.1.2.4 Scaling.

The three-dimensional model generated by the FaroArm is provided in an Stereo Lithography (STL) file format, which is a common format for Computer Aided Design (CAD) and other three-dimensional model representations. Unfortunately, the STL file format does not include specifications for units of measure nor what scale to use. This was discovered during preliminary testing when attempting to compare the CMM model with the Kinect model. Apparently, the FaroArm software stores values in it's STL files using the millimeter scale, where as the Kinect SDK provides values in meters. Not a difficult problem, as a simple scaling factor of 1000 is required to convert the CMM model to the same scale as the Kinect based model for comparison. Scaling the CMM model is preferred, as scaling the Kinect model would reduce the units of precision available in the measurements. This is easily accomplished using the scaling operations included in the available tools, and is simply an extra step to be completed one time.

3.1.2.5 Voxel Per Meter Settings.

For each of the distances chosen as a factor, a model was built at every voxel per meter setting until the KinectFusion algorithm simply could not build a complete model without resulting in significant holes/missing sections of the model airplane, or complete failure of the pose estimation algorithm. For the 0.4 meter distance, at 1408 voxels per meter (vpm), the algorithm was unable to create a complete model, so this setting and any other higher settings were not used for the 0.4 meter distance. This process was repeated for both 0.6 meters and 0.8 meters, and the successful vpm settings for all three distances can be seen in Table 3.1.

3.1.3 Experiment Design.

This experiment follows a full factorial design. For the single Kinect sensor in the survey, there are two factors: distance and resolution(vpm). The sensor distance factor has three levels: 0.4 meters, 0.6 meters, and 0.8 meters. Based on the preliminary testing, the

Table 3.1: Voxel Per Meter Settings. The table below indicates the voxels per meter settings that resulted in succesful model building at each of the chosen distance parameters.

Distance	**Succesful Model Building VPM Settings**							
0.4 m	384	512	640	768	896	1024	1152	1280
0.6 m	384	512	640	768	896	1024	1152	
0.8 m	384	512	640	768	896			

0.4 meter distance has eight values, the 0.6 meter distance has seven values, and the 0.8 meter distance has five values, all of which are shown in Table 3.1. To statistically account for noise and other uncontrollable factors, ten runs (360° scans of the model aircraft) will be recorded, but only three will be used to build the necessary models. This brings the total models built for this experiment to $8 \times 3 + 7 \times 3 + 5 \times 3 = 60$ models.

3.1.4 Performance Metrics.

For the Model Accuracy experiment, the main metric used is the Hausdorff distance as calculated by the CloudCompare tool. The tool will be used to calculate a Hausdorff distance value for each vertex in the Kinect model, which signifies how close that vertex is to it's registered vertex in the FaroArm model. These values are saved by the tool as additional information in the model, and are also used to produce heat maps, which indicate for every portion of the Kinect model, how close it is to it's registered vertex. A statistical analysis, to include the median distance and Interquartile Range (IQR), of the distance values for each vertex make up the entire mathematical comparison between the reference model and the Kinect models. The heat maps provide an additional visual comparison that simply can not be seen in the statistical results alone.

3.1.5 System Parameters.

As mentioned in Section 3.1.3, there are two parameters that will be varied and studied for this experiment. There are also a handful of parameters that will be fixed.

Figure 3.3: Suspended Subject Airplane Model. The 1/48th scale F-22 model is suspended from the ceiling using fishing line, directly over the center of the turn table in Figure 3.1. For the experiment, the aircraft was kept at an approximate 43° nose up, or pitch, angle. This was necessary for the sensor to be able to see both the top and bottom surfaces of the model in a single rotation.

The location of the airplane model is fixed, as the model will be suspended over the center of rotation of the turn table (Section 3.1.1). The model will be suspended from the ceiling using mono filament fishing line, which is small enough to not be detected by the Kinect, so there will be no adverse impact on the model. During preliminary testing, it was determined that a moderate nose-up pitch yielded the best models. For the entirety of the recorded experiment runs, the location (height above the table and in the sensor's field of view) and configuration (approximately 43° nose up pitch angle) will remain the same.

The sensor field of view (Section 3.1.2) is also unchanged for the experiment. The model is positioned near the middle of the sensors field of view, and the tilt of the Kinect motor and the tripod mount are held static for all recordings. Along with the field of view, the sensor's height and the rotation speed of the sensor is static.

As for the KinectFusion algorithm, there are several parameters that affect the model building process. These parameters were explored and tested during the preliminary testing phase. The first parameter determined was the minimum and maximum depth ranges. These two settings allow the system to limit the number of pixels to be processes based on the depth value for that pixel. In this experiment, there is no other object between the sensor and the subject, however, there are objects along the outer edges of the lab area. These objects can be ignored by setting the minimum and maximum depth ranges appropriately. For this experiment, the minimum value is set at 0.35 meters (the default), while the maximum depth range is set to 1.85 meters. This range effectively eliminates any additional objects that may happen to be in the background portion of the field of view of the sensor, and ensures they do not impact the quality of the model. The minimum and maximum depth range settings can be seen at the bottom of Figure 3.4, in the section labeled *Depth Threshold*.

The specific pose estimation method was then selected. The AlignDepthFloatToReconstruction method provides better model tracking results (Section 2.3.3), especially when there are no moving objects in the field of view. With the stationary model and rotating sensor, there are no other objects in the field of view using the depth range limitations, so the AlignDepthFloatToReconstruction method provides the best model quality for this experiment setup. This was confirmed in preliminary testing. As there are only two options for pose estimation methods, the application provides a simple check box labeled *Use Camera Pose Finder*(Figure 3.4-left). With the box checked, the application uses the AlignPointClouds method, and unchecked, will use the AlignDepthFloatToReconstruction method.

According to the SDK documentation [2], the integration weight parameter specifies how tolerant to change the KinectFusion model will be. This value can range from 0 to 1000. Larger values will provide a detailed model but will take longer to average and the model will not adapt to change. A lower value makes the model respond more quickly to

changes in the depth map, but produces a noisier model overall. Based on the findings in the preliminary testing, this is fixed at 500 for the duration of this work. The integration weight setting is set by the slider bar on the right hand side of Figure 3.4, under the label *Volume Max Integration Weight*.

The X, Y, Z values under the *Volume Voxels Resolution* section of Figure 3.4 specify the number of voxels that will be created in each direction, while the vpm setting specifies how much space a voxel will represent. For instance, with X, Y, Z set to 384 voxels, the KinectFusion algorithm will create $384 \times 384 \times 384 = 56,623,104$ voxels. With a setting of 128 vpm, the voxels will represent a cube of volumetric space with each side 3 meters long ($\frac{384 \text{ voxels}}{128 \text{ voxels per meter}} = 3.0 \text{ meters}$). With those settings, a single voxel will represent approximately 476 mm^3 ($\frac{3 \text{ meters}}{384 \text{ voxels}} \times \frac{1000 \text{ millimeters}}{1 \text{ meter}} = 7.8mm$ and $7.8 \times 7.8 \times 7.8 = 476mm^3$).

The precompiled sample application included with the Kinect SDK provides the necessary interfaces for selecting the above described parameters. However, the X, Y, Z sliders are limited to a maximum value of 640, and the maximum vpm setting allowed is 768. Based on the documentation, this is based on the idea that the majority GPU cards on the market will not have the memory required to support more than 262,144,000 total voxels. With the GeForce GTX Titan it is possible to process more voxels, so the sample application was modified to increase the maximum values of the X, Y, Z values to 2048, and the maximum vpm set to 2048.

However, simply using the largest number of voxels possible will create unnecessary additional work for the GPU, which if enough voxels are generated, will cause a slow down and possibly even failure of the model building process. By increasing the X, Y, Z resolution, the volume of space that will be modeled by the KinectFusion algorithm is increased. As the size of the subject is not increased, the added voxels will be empty, or value-less, but will still add to the workload for processing by the algorithm. Instead, the

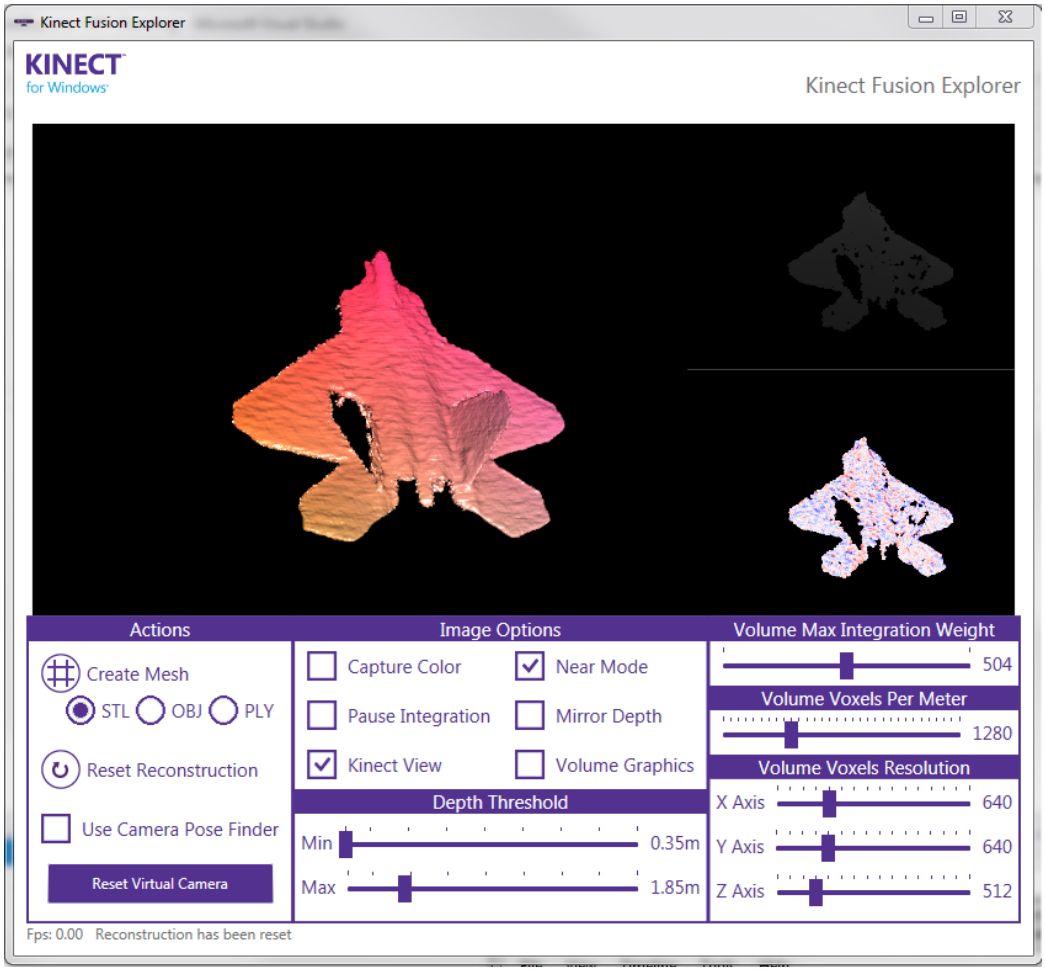


Figure 3.4: Modified KinectFusion Explorer.The sample KinectFusion Explorer application supplied with the Kinect for Windows SDK was modified to increase the maximum Volume Voxels Per Meter setting and the Volume Voxels Resolution for X, Y, and Z. Note that the top right image depicts the current depth view of the Kinect sensor, while the bottom right image depicts the results of the pose estimation method, and the large image in the center is the current reconstruction volume.

X, Y, Z parameters should be set based on the size of the model being scanned and the vpm setting being used.

As discussed in Section 3.1.2.1, the largest dimension of the 1/48th scale model used for scanning in this experiment is 0.3938 meters, which is the length from the tip of the nose to the furthest edge of the horizontal stabilizers at the back of the aircraft. However,

with the model suspended at a 43° pitch angle, the longest dimension of the bounding box becomes approximately 0.36 meters. During a 360° rotation, if any dimension of the volumetric bounding box is less than 0.36 meters, then there is potential for the model to be clipped. An additional 10% is added to the calculated length to ensure no clipping occurs, which brings the desired dimension to 0.396 meters. Due to the interdependence with the voxels per meter setting, the volume voxel resolution setting for X, Y, Z will change as the vpm changes. However, the change is based on Equation (3.1). This can be used to calculate for each vpm setting and X, Y, Z resolution settings what the smallest possible bounding box will be (so as to reduce the number of overall voxels) that will still be larger than 0.396 meters in any dimension. The final X, Y, Z settings used for each vpm setting for each distance can be found in Table 3.2. Note that at the 0.6 meter and 0.8 meter distances, the Z setting was increased in relation to the setting used for the 0.4 meter distances. This was caused by clipping observed during preliminary testing, and was caused by the location of the subject model being a little bit off center in relation to the center of rotation of the turn table. This change in setting does not affect the quality of the model, as any additional voxels that do not contain the subject model will still be hold a value of 0, and does not significantly impact the processing workload of the algorithm.

$$distance = \frac{\text{\# of voxels}}{\text{\# of voxels per meter}} \tag{3.1}$$

3.1.6 Experiment Process.

With the specific settings chosen for each distance factor, experimental runs will be recorded and models will then be built by replaying the recorded session while running the KinectFusion algorithm in the modified software application described earlier. With complete models built for each of the three experiment runs at the three chosen distances, the comparisons between the FaroArm based model and the KinectFusion-based models can be completed. Version 2.5 of the CloudCompare tool will be used to align the

Table 3.2: Volume Voxel Resolution Settings. The table below indicates the volume voxel resolution settings used in the modified KinectFusion application (Figure 3.4) for each vpm setting at the 0.4, 0.6, and 0.8 meter distances. The difference in Z settings at 0.4 meters and 0.6 meters and 0.8 meters is caused by the slightly offset location of the subject model over the experiment turn table. However, the change in settings does not affect the model quality.

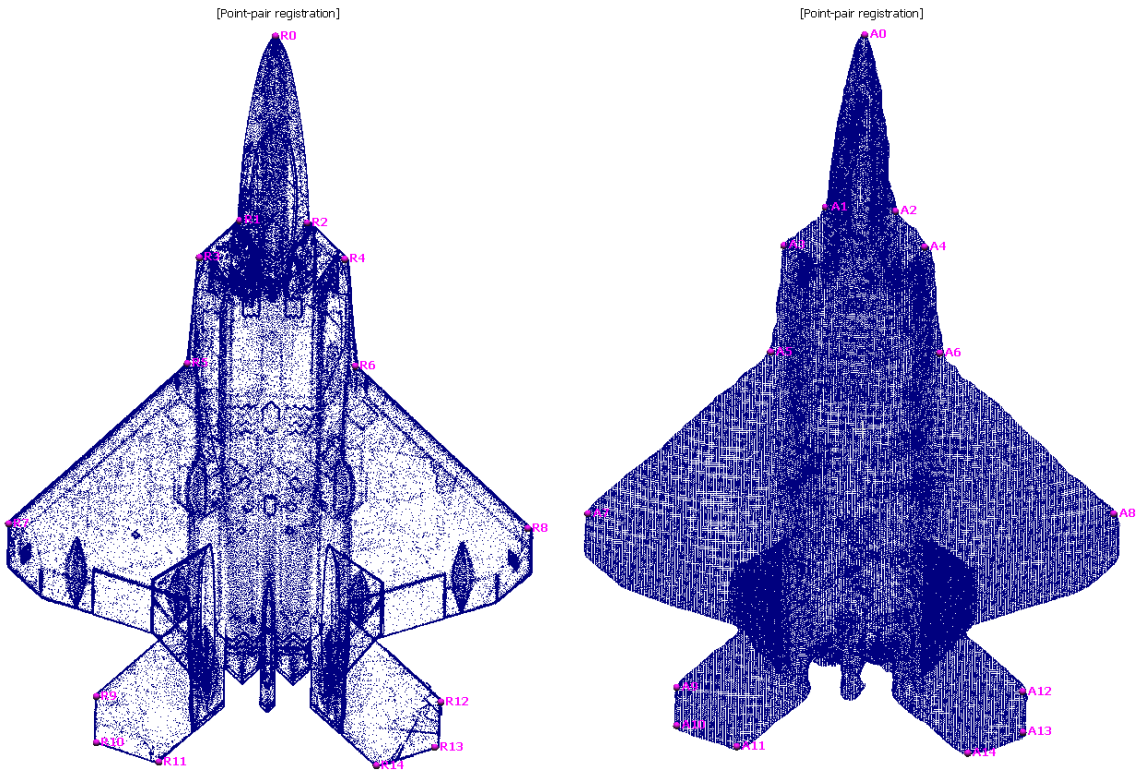
Distance (m)	VPM	X Setting	Y Setting	Z Setting	X Distance (m)	Y Distance (m)	Z Distance (m)	Total Voxels
0.4	384	256	256	256	0.7	0.7	0.7	16777216
0.4	512	256	256	256	0.5	0.5	0.5	16777216
0.4	640	384	384	384	0.6	0.6	0.6	56623104
0.4	768	384	384	384	0.5	0.5	0.5	56623104
0.4	896	384	384	384	0.4	0.4	0.4	56623104
0.4	1024	512	512	512	0.5	0.5	0.5	134217728
0.4	1152	512	512	512	0.4	0.4	0.4	134217728
0.4	1280	640	640	640	0.5	0.5	0.5	262144000
0.6	384	256	256	256	0.7	0.7	0.7	16777216
0.6	512	256	256	384	0.5	0.5	0.8	25165824
0.6	640	384	384	384	0.6	0.6	0.6	56623104
0.6	768	384	384	512	0.5	0.5	0.7	75497472
0.6	896	384	384	512	0.4	0.4	0.6	75497472
0.6	1024	512	512	640	0.5	0.5	0.6	167772160
0.6	1152	512	512	640	0.4	0.4	0.6	167772160
0.8	384	256	256	384	0.7	0.7	1.0	25165824
0.8	512	256	256	384	0.5	0.5	0.8	25165824
0.8	640	384	384	512	0.6	0.6	0.8	75497472
0.8	768	384	384	512	0.5	0.5	0.7	75497472
0.8	896	384	384	512	0.4	0.4	0.6	75497472

two surface models using the ICP algorithm and compute the signed Euclidean distances between the two aligned models. The signed distance values, along with the mean, median, standard deviation, and other statistical measurements on the surface reconstructions will then used to help quantify the quality of a given model in comparison to the *ground-truth* model.

In order for the CloudCompare tool to compute the signed distances, the two models must go through a two phase alignment process. The first phase is a general (coarse) alignment, which is accomplished by the user in one of two ways. The first is to use the built in transformation tool to apply the necessary rotation and translation operations to one model in order to align it with the second model. The second method requires the user to pick at least three points on one model, and then pick three points on the second model that are the closest to the original points. For the point picking method, the more points picked will yield a better coarse model alignment.

This point picking method will be utilized for each model, and there are 14 specific points that will be utilized on the subject model to ensure a proper course fitting. In the CMM based model there is plenty of detail in order to differentiate between specific sections of the subject model. However, in the KinectFusion based models, this same level of detail is absent, so care was taken to find the best points that would be as distinct as possible, no matter the quality of the model. To this end, the best points were found to be at the sharp corners of the aircraft models. These points include the very tip of the nose, the shoulder points where the canopy meets the rest of the fuselage, the ends of the wingtips, and the three corners on each of the horizontal stabilizers. These 14 points can be seen in Figure 3.5a and Figure 3.5b.

Following the coarse alignment, the ICP algorithm is applied to fine tune the registration. In the CloudCompare tool there are two parameters that govern the quality of alignment that can be achieved with the ICP algorithm. The error difference parameter



(a) FaroArm Model (b) Kinect Model

Figure 3.5: Picking Points for Course Mesh Alignment. Note that the sharp corner points were used, as these areas are the most distinguishable at even the lowest resolution Kinect models.

is the setting for the threshold at which the iterations of the algorithm will stop once the error distance between models is below the threshold. According to the CloudCompare documentation for applying the ICP algorithm [1], 1e-8 is the near the computational accuracy limit of the tool, so 1e-10 will be used for this work to ensure the best possible alignment. The second parameter, random sampling limit, specifies the number of points in each model to be used for the calculations. If the value is set higher than the number of points available in a model, there will be no limit and the entire collection of points will be used for the calculation. For this work, a random sampling limit of 400,000 will be used, as this is larger than the number of points in the largest model.

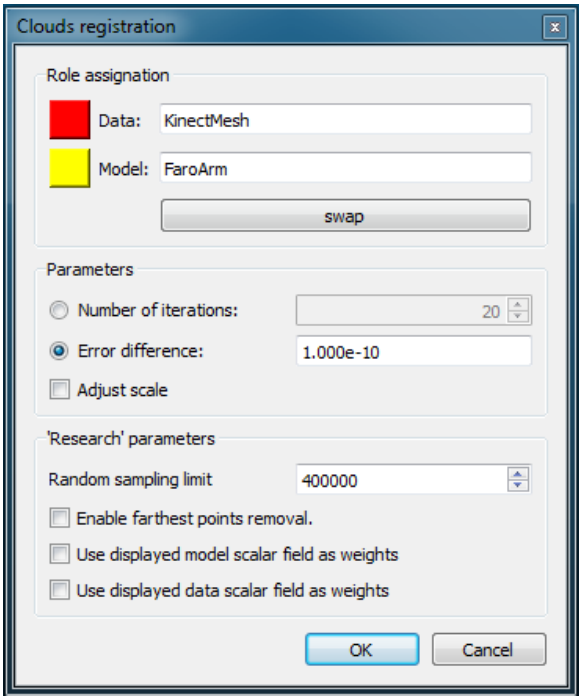


Figure 3.6: CloudCompare Fine Alignment. This is the dialog box and the used settings for the fine model alignment in CloudCompare. The error difference specifies when the ICP algorithm should terminate, and the random sampling limit dictates how many samples of the model to use. In this case, 400000 is used to bypass the sample limitation, as the models of used for this experiment are not very large and do not require lengthy computations.

Once the two models have been aligned, the Hausdorff distance metric can be computed for each point in the KinectFusion based model. The CloudCompare tool utilizes a two step process for calculating the distance between two point clouds. The first step is a quick and imprecise calculation that will help determine the necessary parameters to be used by the second step. The imprecise process is based on the chamfer distance metric, which computes the distance between every point on an edge in one model to its nearest edge point in the second model, and sums the total distance across all the edge points. Chamfer has been shown to be computationally fast, but is considered imprecise.

With the Chamfer distance calculation complete, the parameters for the precise distance calculations are set automatically, though they are configurable by the user.

For this experiment, the automatic parameters determined by the Chamfer distance computation will be used for the precise calculations. The dialog box for the distance computation is seen in Figure 3.7, which shows the imprecise distance calculations and the automatic selection of parameters. The precise distance computation is based on the Hausdorff distance metric discussed in Section 2.3.4. In version 2.5 of CloudCompare, the distance calculations can be performed either using or ignoring signed distances. For this experiment, signed distances will be used, where positive values indicate points of the KinectFusion model that are above their matching point in the FaroArm model, and negative values indicate where the KinectFusion model is below the FaroArm model.

In the console area of the CloudCompare tool, a report of the Hausdorff distance calculation is provided which indicates the mean distance metric and the standard deviation of all the distances for the particular model (Figure 3.8). These values form the basis for comparing one Kinect model to another. After calculation, CloudCompare can be used to export the calculated distance for each point in the model, which will then be used for additional statistical analysis.

After computing the Hausdorff distance for every point in the KinectFusion model's point cloud a *heat map* can be created to visually indicate where the KinectFusion model varies from the FaroArm model. The specific color indicates just how much the models vary at a given location (Figure 3.9). A histogram representation along the scale illustrates the distribution of error distances throughout the entire model. A finer detailed histogram can also be obtained in CloudCompare, and is seen in Figure 3.10. For this work, both the statistical analysis of the calculated distances and a visual comparison of the generated heat maps will be used to quantify the quality of the KinectFusion algorithm's three-dimensional modeling capabilities. All of the heat maps and histograms for this work will be presented with values in millimeters.

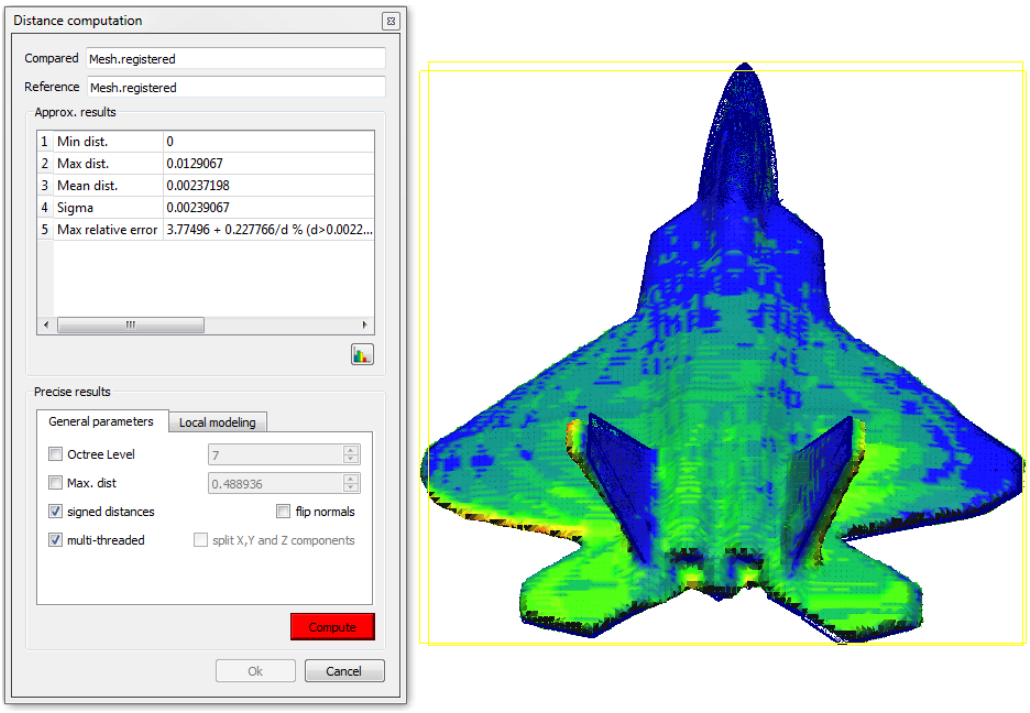


Figure 3.7: Distance Calculation. Initial alignment and parameters are computed by CloudCompare using the Chamfer distance metric. Based on those calculations, the system automatically sets the parameters under "General parameters" though they can be modified by the user. For this experiment, the system generated parameters were utilized. Signed distances were utilized to indicate where the surface of the KinectFusion model is *above* or *below* the surface of the FaroArm model. The coloration of the model in this figure is based on the Chamfer distance colorizations values, and is replaced by the Hausdorff distance values after the computation is complete.

```
[10:16:21] [ComputeDistances] Time: 0.81 s.
[10:16:21] [ComputeDistances] Mean distance = 0.002189 / std deviation = 0.003050
```

Figure 3.8: CloudCompare Distance Calculation Summary.This is the summary message presented after the calculation of the distance values for the entire model. While the mean and standard deviation are good for a quick comparison, the full collection of distances must be exported by CloudCompare for further statistical analysis.

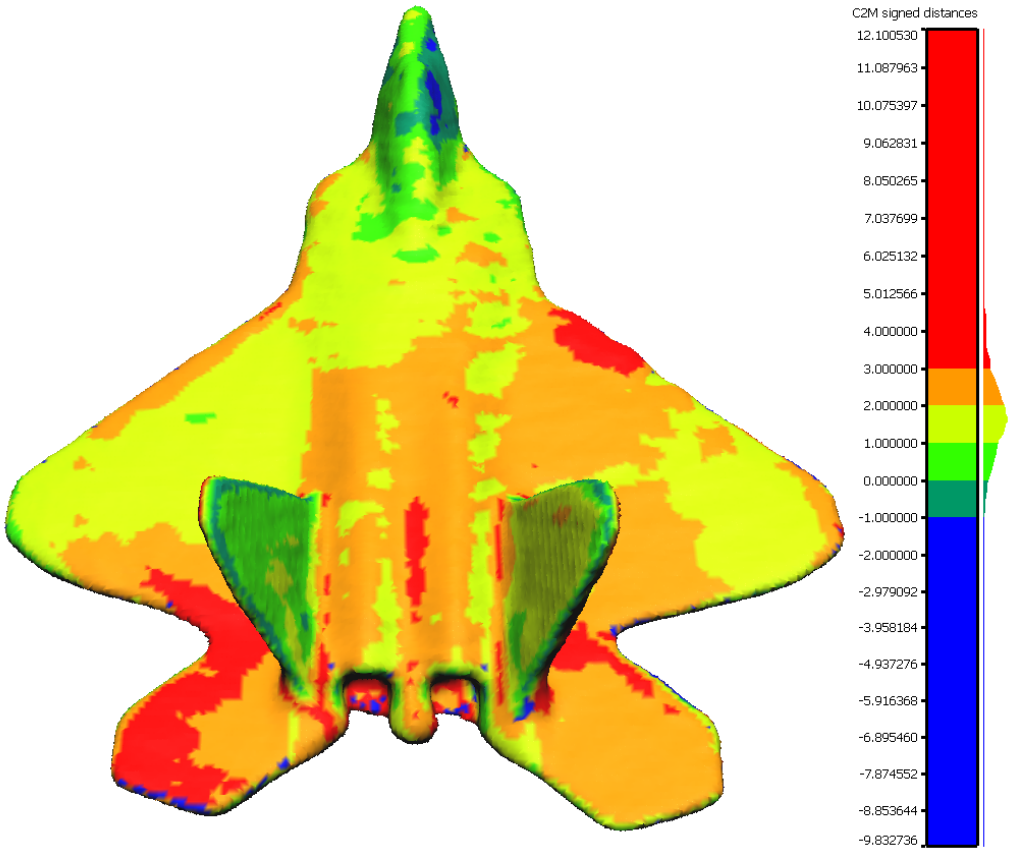


Figure 3.9: Sample Distance Heat Map. This is a sample of a heat map created based on the Hausdorff distance values at each vertex of the Kinect based model. The colors indicate the magnitude of the distance between models, and the scale is seen on the right side. Along the scale is also a histogram representation of the distribution of Hausdorff distances across the entire model. All values are in millimeters.

3.2 Quantification of Anomaly Detection

This experiment will provide a second quantification of the accuracy and precision provided by the KinectFusion algorithm. It will help determine just how large objects must be in order to be detected in a three-dimensional model built using the KinectFusion algorithm. Along with detection capability, it will also quantify how well those objects are modeled in comparison to the true physical dimensions of the objects.

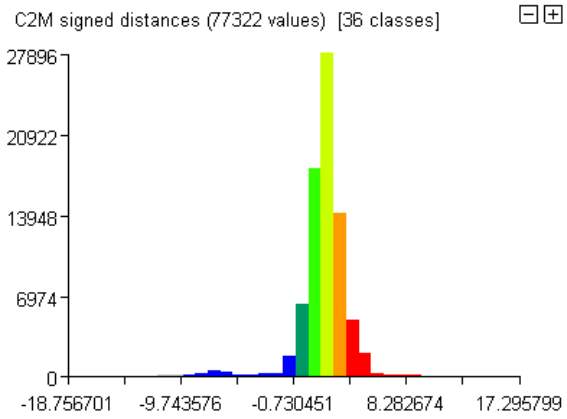


Figure 3.10: Sample Distance Histogram. This is a sample of the detailed histogram available in CloudCompare after calculating the distances between two models. The horizontal axis scale is in millimeters. The vertical axis scale is in number of samples. This sample histogram indicates that the distribution of error distances is tightly grouped slightly above 1 mm.

3.2.1 Experiment Setup.

In order to quantify the minimum surface deviation detection capability of the KinectFusion algorithm, a slightly different experiment was needed. With the F-22 setup (Section 4.1), adding noticeable defects would require a scaling factor to match the F-22. Unfortunately, adding small items such as dimes to the surface of the 1/48th model would be in fact similar to adding an object 48 times the size of a dime, which is quite easily seen in a visual inspection, and would not provide a true quantification of the KinectFusion's ability to discern minuscule items. To that end, a different experiment setup was required.

For this experiment, a model will be created of a basic scene consisting of a flat table top and stacks of US coins placed on the table surface. The stacks of coins provide objects of varying thickness (or height) and width for modeling, and will help determine the KinectFusion's minimum threshold for modeling ability. The stacks will provide objects as small as 1.3 mm in height to as large as 12 mm in height [3]. Table 3.3 shows the diameter and thickness of the US penny, nickel, dime, and quarter in multiples of one to six. As 0.4

Table 3.3: United States Coin Physical Dimensions. The below values are the physical size dimensions, diameter and thickness, of the penny, dime, nickel, and quarter coins [3].

Coins	1		2	3	4	5	6
Coin	**Diameter (mm)**	**Thickness (mm)**	**Thickness (mm)**	**Thickness (mm)**	**Thickness (mm)**	**Thickness (mm)**	**Thickness (mm)**
Nickel	21.21	1.95	3.90	5.85	7.80	9.75	11.70
Quarter	24.26	1.75	3.50	5.25	7.00	8.75	10.50
Penny	19.05	1.55	3.10	4.65	6.20	7.75	9.30
Dime	17.91	1.35	2.70	4.05	5.40	6.75	8.10

meters is the closest possible operating distance, this experiment will only be conducted at the 0.4 meter distance, and at the highest voxels per meter setting possible, 1280 voxels per meter (as discussed in Section 3.1). The MeshLab tool will then be used to measure the modeled version of the coins, which will be compared to their real life counterparts. The aim of this comparison is to quantify the accuracy of the KinectFusion algorithm.

Similar to the model accuracy experiment, in order to build a complete model of the coin stacks, the sensor must be rotated around the coins. Unfortunately, the turn table setup used for the model accuracy setup can not be used since the sensor is mounted to a support arm that is mounted to the turn table which must sit on the floor. Because of the rotating arm, a table can not be placed over the turn table for simple 360° scanning. Instead, the sensor will be moved by hand, in a similar manner for each model building session. For this experiment, the sensor will start at the front of the table, at approximately a 30° angle. The sensor will then be moved, from front to back and back to front, over the surface of table, maintaining a 0.4 meter distance. Once returned to the beginning position, the sensor will be rotated approximately 150° to both the left and right of the table, again, maintaining a 0.4 meter distance from the nearest coin stack.

3.2.2 Preliminary Testing, Results, and Findings.

A simple preliminary test was conducted to determine the suitability of this experiment and to solidify details required for the final experiment run. During the test, stacks of

varying numbers of US coins were placed in rows on a solid surface, and then modeled using the KinectFusion algorithm. Unlike the preliminary tests for the Model Accuracy experiment, there were no significant issues found during testing.

3.2.3 Experiment Design.

This experiment follows a simplistic design. There is only one operating distance (0.4 meters), one voxels per meter setting (1280 vpm), and to account for noise and other uncontrollable factors, seven runs will be recorded, and four will be chosen at random to build models from. There will be 24 stacks of coins on the table, with stacks of one, two, three, four, five, and six of each of the penny, nickel, dime, and quarter.

3.2.4 Performance Metrics.

The metrics utilized for this experiment will consist of a binary stack detection and an error calculation. For the stack detection, either the stack of coins was detected and modeled by the algorithm, or it wasn't. If the stack of coins was detected, then a modeling error metric can be calculated. This will done by measuring the thickness of the modeled stack of coins and comparing those values with the known values of the physical coins. The difference between the modeled values and the physical values is the modeling error.

3.2.5 System Parameters.

For this basic experiment, there will be no varied parameters and only fixed parameters. As discussed earlier, the distance between the sensor and nearest stack of coins will be maintained at 0.4 meters. Given the model size issues found in Section 3.1.2.1, it is expected that the minimum detection level will decrease as the sensor distance increases. This experiment is looking for the minimum overall, so utilizing the best possible distance and no other value is appropriate.

The sensor field of view, as with the model accuracy experiment, will be fixed for the duration of the experiment. Along with field of view, the speed of rotation/movement will also be fixed. For this experiment, the sensor is held in hand and slowly moved around the

scene in order to capture all viewing angles. To the extent that is humanly possible, the speed of rotation will be considered fixed for this experiment.

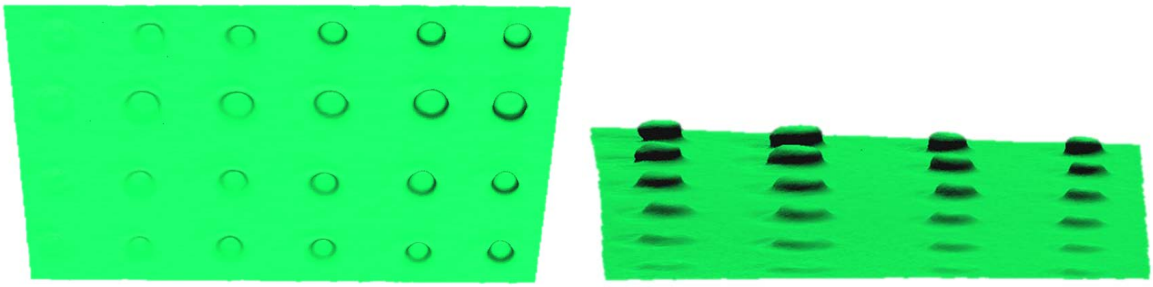
The volume voxel resolutions settings dictate the X, Y, Z values of the volumetric box in three-dimensional space that will be modeled using the KinectFusion algorithm. Along with the voxels per meter parameter, these parameters will dictate the physical size of the volumetric box. Based on the results seen in preliminary testing, the values were set at 512 for X, Y, Z. As mentioned, the highest possible voxels per meter setting that provided successful models in the model accuracy experiment, 1280, will be used for this experiment. As in the model accuracy experiment, the integration weight will be left unchanged at 500.

Finally, unlike the model accuracy experiment, the AlignPointClouds method for pose estimation will be used. In preliminary testing, the AlignPointClouds method yielded better modeling results, as the AlignDepthFloatToReconstruction method failed on several modeling attempts.

3.2.6 Experiment Process.

As discussed earlier, stacks of one to six of each of the pennies, nickels, dimes, and quarters will be assembled on a flat surface. The six stacks for each coin will be placed in their own row, with one of each coin on the far left and six of each coin on the far right. The rows will be aligned with the shortest thickness, the dimes, at the front, and the largest thickness, the nickel, at the rear. The stacks in all rows and columns will be spaced approximately 35 mm apart, to allow for sufficient distance between stacks such that the shadow of one stack will not obscure any other stack.

With the stacks of coins setup, the sensor will be moved as described in Section 3.2.1. For this experiment, seven independent experiment runs will be recorded at the 0.4 m distance, of which four will be picked arbitrarily to be modeled and analyzed. Each of the four runs will then be modeled using the same modified software as used in the



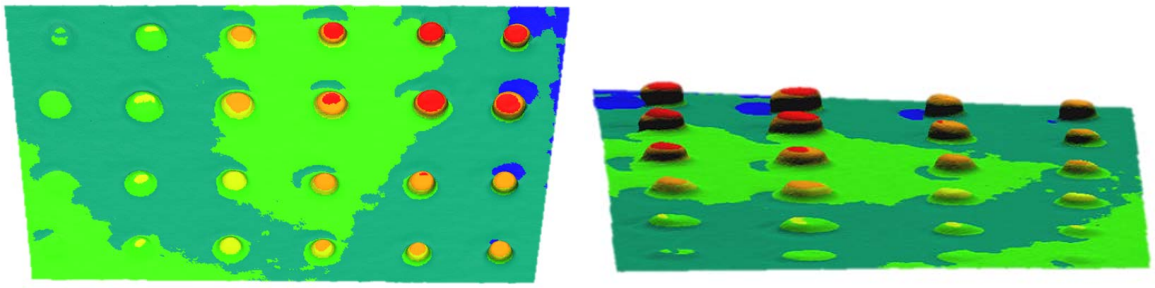
(a) front view (b) side view

Figure 3.11: Coin Models. The above figures show one of the coin models before distance calculations. Note that in (a) the single dime is at the bottom left, and the stack of six nickels is at the top right. A rotated view of the same model is seen in (b), looking down the rows of coins, with the nickels on left and dimes on the right side. The single stack of each coin is barely visible in (a) but more visible in (b).

Section 3.1. The models will then be edited to remove any extraneous items, leaving just the planar surface and the stacks of coins. A sample of the completed models can be seen in Figure 3.11.

Initially, the same analysis procedures as Section 3.1.6 were used to calculate the height at the top of each stack of coins. CloudCompare was used to create a plane similar in size to the planar surface in the coin model. From there, the two planar surfaces were coarsely and finely aligned using the same methods as used in the model accuracy experiment. With the models aligned, the distance values could then be computed in the same manner as before. The resultant heat maps can be seen in Figure 3.12.

Unfortunately, as shown in Figure 3.12, the flat surface does not align perfectly with the true plane. This is due the noise of the Kinect sensor, which causes points on the same flat surface being modeled to have different depth distances, which when added to a dense surface reconstruction, are modeled as numerous triangles, each with slightly varying surface normals, which creates an inconsistent surface. Because of the inconsistent surface,



(a) front view (b) side view

Figure 3.12: Coin Model Heat Maps. The above figures show the heat maps of one of the coin models. Note that in (a) the single dime is at the bottom left, and the stack of six nickels is at the top right. A rotated view of the same model is seen in (b), looking down the rows of coins, with the nickels on the left and dimes on the right side. The single stack of each coin is barely visible in (a) but slightly more visible in (b).

the height calculations for the top of the stack of coins varied significantly between model runs, which led to inconsistent results.

In order to ensure consistent measurements, the height of each stack will be measured with respect to a localized plane around the subject stack of coins. To define a localized plane for each stack of a coins, three points near the stack, but not on the stack, must be selected. Equation (3.2) defines the general form equation for a plane, where a, b, and c can be determined by three non-collinear points, A, B, and C, in three-dimensional space. Since the three points lie in the plane, they can be used to form two vectors in the plane, which can then be used to define a normal vector to the plane by computing the cross product of the two vectors. Vectors AB and AC are defined by Equations (3.3) and (3.4) respectively. Values for a, b, and c can be found from the cross product of vectors AB and AC, and are given by Equations (3.5), (3.6), and (3.7). Finally, d can be found using any of the three points (A, B, or C) and is given by Equation (3.8) using point A as an example. Next, many points on the top of the coin stack will be selected to characterize the height distribution of each stack of coins. With the equation of the localized plane defined, the distance between

any three-dimensional point P on the top of the coin stack and the plane can be found using Equation (3.9)[21].

$$ax + by + cz + d = 0 \tag{3.2}$$

$$AB = (B_x - A_x, B_y - A_y, B_z - A_z) \tag{3.3}$$

$$AC = (C_x - A_x, C_y - A_y, C_z - A_z) \tag{3.4}$$

$$a = (B_y - A_y)(C_z - A_z) - (C_y - A_y)(B_z - A_z) \tag{3.5}$$

$$b = (B_z - A_z)(C_x - A_x) - (C_z - A_z)(B_x - A_x) \tag{3.6}$$

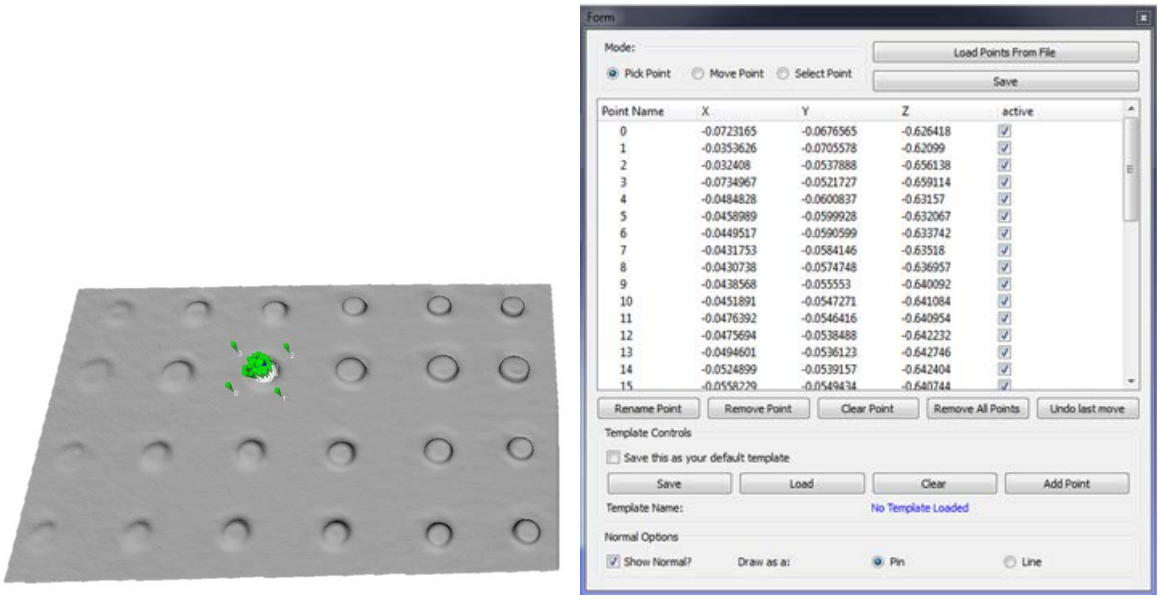
$$c = (B_x - A_x)(C_y - A_y) - (C_x - A_x)(B_y - A_y) \tag{3.7}$$

$$d = -(aA_x + bA_y + cA_z) \tag{3.8}$$

$$distance = \frac{|aP_x + bP_y + cP_z|}{\sqrt{a^2 + b^2 + c^2}} \tag{3.9}$$

With a new method determined, a second tool, MeshLab, will be utilized to pick points on the flat surface and the stack of coins. A view of MeshLab's point picking interface and selected points in the model can be seen in Figure 3.13.

For each stack of coins in each model, the following procedure in MeshLab will be utilized. The first four points selected will be on the flat surface and will be used to define four localized planar surfaces for measuring height against. By selecting four points instead of just three, four different planes can be defined and used for height calculations. The variation of the planes can then be averaged out to provide a more accurate height measurement. After the four plane defining points picked, at least 30 points will be selected that lie on the top surface of the coin stack. The 30 points will be evenly distributed across the top surface, but since they will be chosen by the user, they are not truly random. Despite not being random, the 30 points provide a good distribution of samples for accurately measuring the height and variation of the modeled coin stack.



(a) front view

(b) side view

Figure 3.13: MeshLab Point Picking. The green inverted pyramids in (a) represent the normal vector of the selected points. The coordinates of each point are listed in the table in (b), where the values are currently shown in meters. These values are later scaled to provide values in millimeters.

The four plane points for each coin stack will be used to define the equations for four localized planes. For each plane, the 30+ height points for each coin stack will then used to calculated the distance between the localized plane and the height point. This will produce at least 120 height measurements for each of the 24 stacks of coin in each model. The measurements for each coin stack will then be combined with measurements for the same coin stack in the other three models. The statistical analysis will then be conducted on the combined values.

IV. Results and Analysis

This chapter presents the results of two experiments: the model accuracy experiment comparing the CMM based reference model and the surface reconstruction models constructed by the Kinect sensor and the KinectFusion algorithm (Section 3.1), and the quantification of anomaly detection experiment (Section 3.2), which modeled sets of US coins to measure the degree to which the KinectFusion algorithm can detect objects of very small dimensions.

4.1 Model Accuracy Experiment Results

For this experiment, ten runs, each consisting of a full 360° rotation around the suspended model, were recorded at each of the prescribed distances. The analysis here is based on three of the ten runs at each distance (selected at random). Specifically, runs 2, 4, and 6 were selected at the 0.4 meter range, while runs 5, 6, and 9 were selected at the 0.6 meter range, and runs 1, 6, and 10 were selected for the 0.8 meter range.

4.1.1 Quality vs Voxels Per Meter.

In order to begin to quantify the quality of models in this experiment, more than just the mean and standard deviation of the distance values were needed for analysis, so all of the distance values for each model were exported for further analysis. For each model, the arithmetic mean, standard deviation, maximum, minimum, median, range, third quartile, first quartile, and IQR were calculated. Along with the statistics for each individual model from the three separate experiment runs, the distance values for each model at each vpm setting were combined and the same statistics were calculated for the combined values. Table 4.1 shows the statistics for the models generated at 0.4 meters over the varying vpm settings. The results show that there is an improvement in the accuracy of the KinectFusion based model as the voxel resolution setting is increased. For example, at the lowest voxel

Table 4.1: Statistical Analysis of 0.4 Meter Experimental Runs. The below values are the statistical analysis of the combined experimental model building runs for each of the vpm settings at the 0.4 meter operating distance.

Voxels Per Meter	384	512	640	768	896	1024	1152	1280
Samples	85698	155265	237747	338083	461100	649210	795742	932715
Mean (mm)	2.3	1.5	0.8	0.5	0.2	-0.5	-0.6	-0.6
SD (mm)	3.3	2.9	2.9	2.7	2.5	3.1	3.0	2.4
Maximum (mm)	22.2	18.1	15.7	15.2	15.4	15.5	15.4	10.3
3rd Quartile (mm)	4.4	3.3	2.2	1.9	1.4	1.1	0.9	0.7
Median (mm)	2.2	1.4	1.0	0.6	0.4	0.0	0.0	-0.2
1st Quartile (mm)	0.2	-0.2	-0.4	-0.4	-0.7	-1.2	-1.4	-1.5
Minimum (mm)	-13.6	-18.1	-16.3	-16.8	-16.5	-17.4	-17.4	-13.5
IQR (mm)	4.2	3.5	2.6	2.2	2.1	2.3	2.3	2.1

setting used, 384 vpm, the median distance error value was 2.244 mm (IQR=4.212 mm), while the median distance value of the 1024 vpm models was 0.029 mm (IQR=2.283 mm), which is a 98% decrease in the distance between the models. However, a decrease in median distance alone does not signify an increase an accuracy. The continual decrease in the IQR, which signifies a decrease in the level of uncertainty in the model, combined with the decrease in the median distance error indicates an increase in model accuracy and precision.

However, one of the more interesting aspects of the data in Table 4.1 is the notion of a point of diminishing returns. In the metrics used to quantify the quality, there are also specific points at which they increase slightly (indicating an increase in distance and thus a decrease in quality), and then decrease again. The changes in the three metrics is highlighted in Figure 4.1. For example, the median value decreases steadily from 384 vpm to 1024 vpm, and then continues to decrease on the negative side, indicating that the later

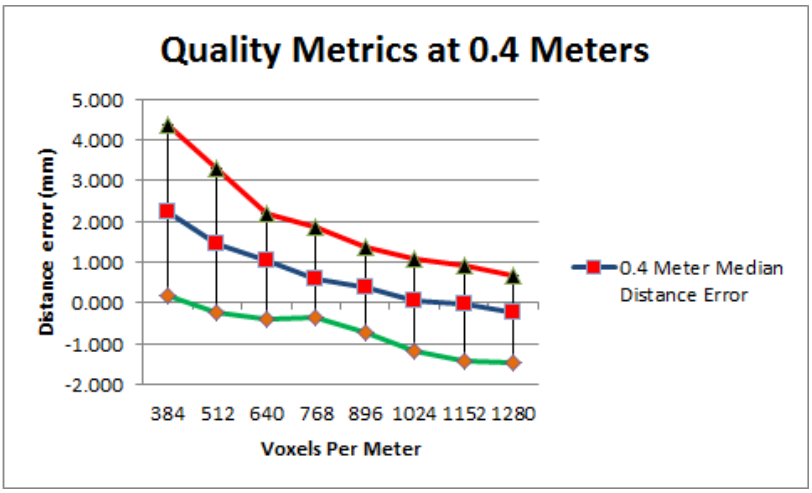


Figure 4.1: Quality Metrics at 0.4 Meters. The median distance error for the given voxels per meter settings at the 0.4 meter operating distance. The upper line indicates the top of the IQR, or 75th percentile, while the bottom line indicates the bottom of the IQR, or 25th percentile. There is both a steady decline in the median value, as well as a tightening of the IQR which indicates less uncertainty, and thus a better accuracy and quality model at higher voxels per meter settings.

models are again increasing the distance between the FaroArm and Kinect based models, but with the Kinect models now being less than/shorter than the FaroArm ground truth model. Secondly, in the standard deviation metric, a steady decrease is seen from 384 vpm down to 896 vpm, but then a significant increase in standard deviation at 1024 vpm, followed again by a steady decrease and the final value at 1280 vpm being just under the standard deviation at 896 vpm. A similar pattern is seen in the IQR. All together, this may indicate that between 896 and 1280 vpm, there may be a point at which the model will no longer get better in quality, and any further increases in vpm will be pointless.

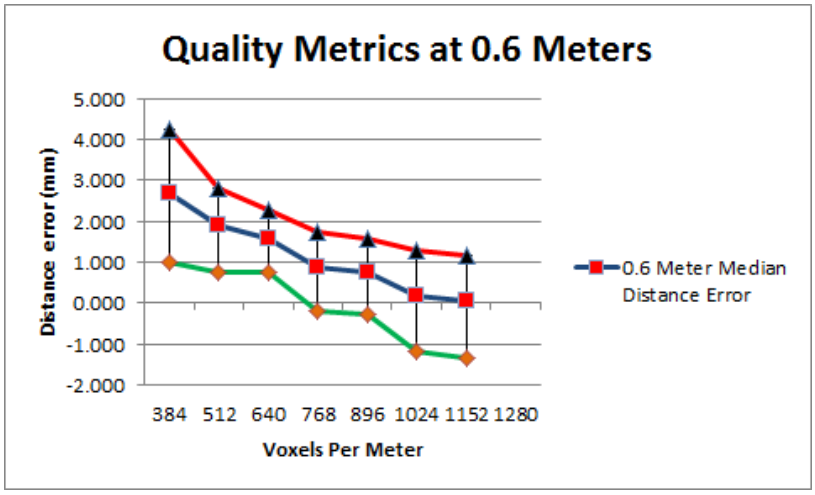
Similar findings are also present in the statistical analysis of the model data at 0.6 meter and 0.8 meter operating distances, in Table 4.2 and Table 4.3 respectively. The trends for the metrics for both 0.6 meters and 0.8 meters can be found in Figure 4.2a and Figure 4.2b respectively. In all of the distance cases, the median distance value decreases

Table 4.2: Statistical Analysis of 0.6 Meter Experimental Runs. The below values are the statistical analysis of the combined experimental model building runs for each of the vpm settings at the 0.6 meter operating distance.

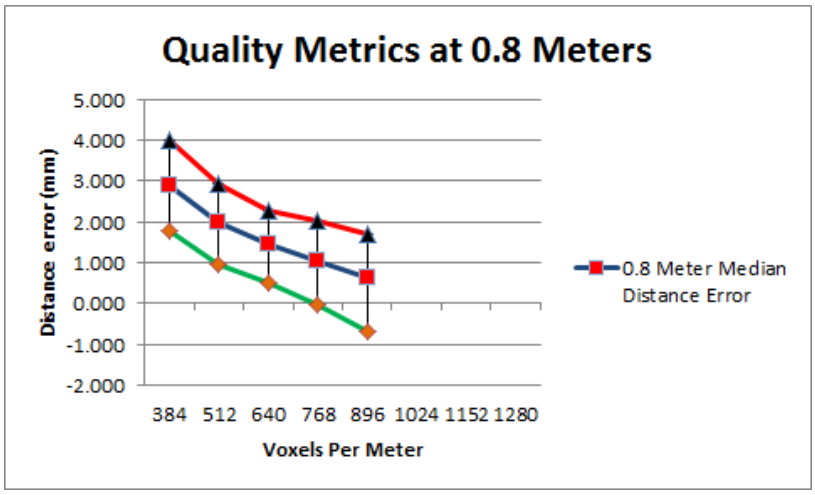
Voxels Per Meter	384	512	640	768	896	1024	1152
Samples	86505	156865	233731	346016	459219	632869	811276
Mean (mm)	2.6	1.4	1.3	0.4	0.3	-0.4	-0.6
SD (mm)	2.6	3.0	2.1	2.7	2.5	3.1	3.0
Maximum (mm)	15.6	13.6	12.0	13.6	13.0	15.4	17.3
3rd Quartile (mm)	4.2	2.8	2.3	1.7	1.6	1.3	1.1
Median (mm)	2.7	1.9	1.6	0.9	0.7	0.2	0.1
1st Quartile (mm)	1.0	0.8	0.7	-0.2	-0.3	-1.2	-1.4
Minimum (mm)	-16.4	-18.1	-16.2	-17.5	-16.8	-17.3	-18.8
IQR (mm)	3.2	2.1	1.5	1.9	1.8	2.5	2.5

steadily from 384 vpm through 896 vpm. Unfortunately, the decreasing trends seen in the 0.4 meter IQR are not seen in the 0.6 meter and 0.8 meter models. In the 0.6 meter chart, there are decreases in the IQR followed increases. In the 0.8 meter chart, the IQR remains fairly constant.

Unfortunately, trying to quantify the quality of a model based solely on a single numerical value is nearly impossible. The metrics discussed and seen above provide a snap shot, but do not tell the complete story of the accuracy of the models. Heat maps based on the error distances, on the other hand, provide a quick and effective means of portraying the quality of the model over the entire surface of the reconstruction. Figure 4.3 provides the top and bottom views, as well as the histogram distribution of the distance



(a) 0.6 meter Metrics



(b) 0.8 meter Metrics

Figure 4.2: Quality Metrics at 0.6 and 0.8 Meters. The median distance error and interquartile range for each voxels per meter setting at both 0.6 meters (a) and 0.8 meters (b). The top line indicates the top of the IQR range, or 75th percentile, while the bottom line indicates the bottom of the IQR range, or 25th percentile. Both figures show a steady decline in the median distance error value as the voxels per meter setting increases. However, the IQR is erratic at 0.6 meters, decreasing and then increasing, and then decreasing again, and while the IQR at 0.8 meters is fairly steady as the voxels per meter setting increases.

Table 4.3: Statistical Analysis of 0.8 Meter Experimental Runs. The below values are the statistical analysis of the combined experimental model building runs for each of the vpm settings at the 0.8 meter operating distance.

Voxels Per Meter	384	512	640	768	896
Samples	87487	151862	234395	335348	454807
Mean (mm)	2.7	1.6	1.0	0.7	0.2
SD (mm)	2.6	2.7	2.7	2.6	2.7
Maximum (mm)	15.5	15.4	15.5	14.8	15.1
3rd Quartile (mm)	4.0	2.9	2.3	2.0	1.7
Median (mm)	2.9	2.0	1.4	1.0	0.6
1st Quartile (mm)	1.8	1.0	0.5	0.0	-0.7
Minimum (mm)	-16.9	-16.9	-17.4	-17.7	-17.3
IQR (mm)	2.3	2.0	1.8	2.0	2.4

errors for the entire model at three different voxel settings, 384, 896, and 1280 at the 0.4 meter operating distance.

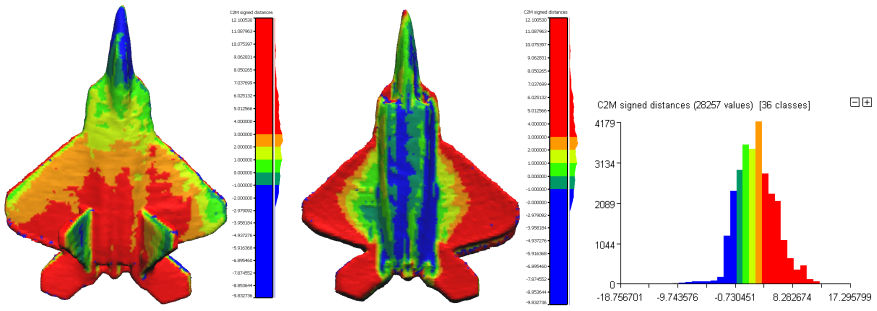
The same observations that were made from the tables and charts for the 0.4 meter distance can also be made from the heat map representations. For instance, the histograms highlight both the decline in the median distance error value, as well as the tightening of the IQR, or uncertainty, band as the vpm setting is increased. At 384 vpm, a large portion of the aircraft is red, which indicates error distances greater than 3.0 mm. There is much less red on the 896 vpm, and even less at the 1280 vpm models. However, there is also more blue at 1280 voxels, indicating the KinectFusion model is *inside*, or below, the FaroArm based model.

Based on the information within Table 4.1 and Figure 4.1, one could argue that the best KinectFusion model at the 0.4 meter distance occurs at the 1280 vpm setting. However,

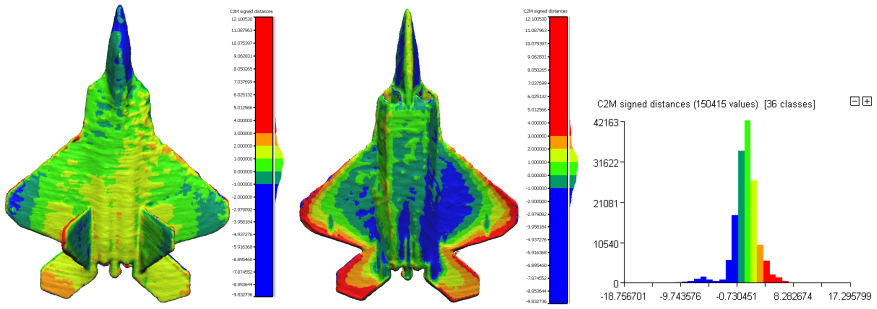
looking at the amount of dark blue (distance errors greater than -1.0 mm) in the respective heat maps, one could argue that the 1280 vpm model is actually worse than the 1024 or even 896 vpm models. To support that argument, the heat map for 896 vpm (Figure 4.3d and Figure 4.3e) includes large amounts of the model with error distances between 0 and 1.0 mm. The histograms for both 896 vpm (Figure 4.3f) and 1280 vpm (Figure 4.3i) show very similar median distance error values, and very similar widths in the distribution functions. More importantly, the amount of blue shaded bars in the 1280 vpm histogram highlight that a larger portion of the 1280 vpm model is less than the reference model. This information indicates that increasing the number of voxels in a reconstruction, when paired with the Kinect sensor, is not always going to provide an improvement.

Apart from confirming the observations that were formed by analyzing the tables and charts, the heat maps provide insight in to other aspects that affect the quality of the models. The largest of which is the *voxelation* of the KinectFusion based models, especially at lower voxel settings. The term voxelation is used here to refer to the symptoms similar to pixelation in two dimensions, where the individual voxels are so large that they present a *blocky* representation of the model. This issue can be seen in the wings of the KinectFusion models in the side view at each of the voxel settings (Figure 4.4). As expected, as the number of voxels is increased by the vpm setting, the size of an individual voxel decreases, which decreases the voxelation of the model.

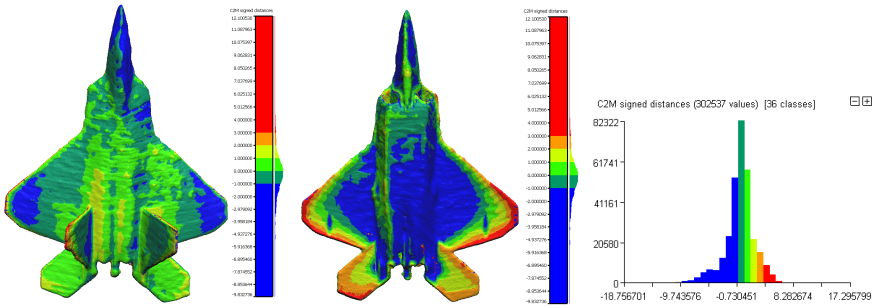
Along with simply decreasing the voxelation, there is also an improvement in detection of smaller features on the model aircraft. In Figure 4.5a there is no indication that there is anything but a semi-smooth surface on the underside of the wings. However, in Figure 4.5b, two small items on the underside of the wings begin to emerge as the wings begin to thin out with the increase in voxels. These two items, along with two smaller items that do not appear in the KinectFusion models, can be seen in the FaroArm based model



(a) 384 vpm - Top View (b) 384 vpm - Bottom (c) 384 vpm - Histogram



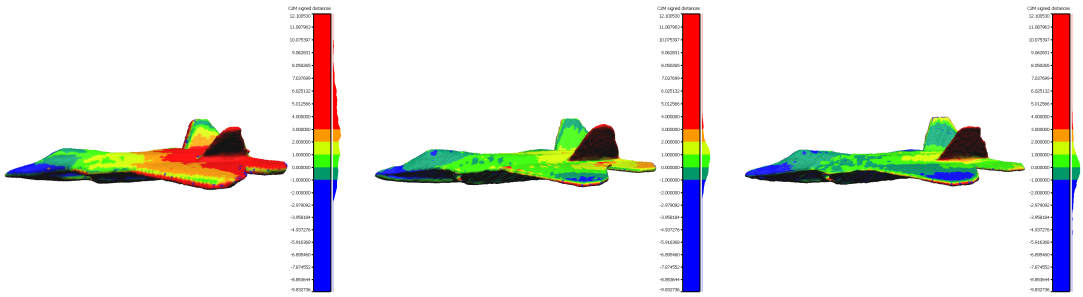
(d) 896 vpm - Top View (e) 896 vpm - Bottom (f) 896 vpm - Histogram



(g) 1280 vpm - Top View (h) 1280 vpm - Bottom (i) 1280 vpm - Histogram

Figure 4.3: Visual Comparison of Varying Voxels Per Meter @ 0.4 Meters. The top and bottom views, along with the histogram depicting the distribution of distance error values for the KinectFusion based models at various vpm settings at the 0.4 meter operating distance. Based on the shading, the 384 vpm model is significantly larger than the reference model, and thereby a lower quality model than then 896 vpm model. Additionally, the 1280 vpm may well be worse than 896 vpm model based on the amount of dark blue shading in the two models. All heat map values and histogram values are in millimeters.

in Figure 4.5c. The heat map representations for all of models created at the various vpm settings and operating distances can be found in Appendix A.



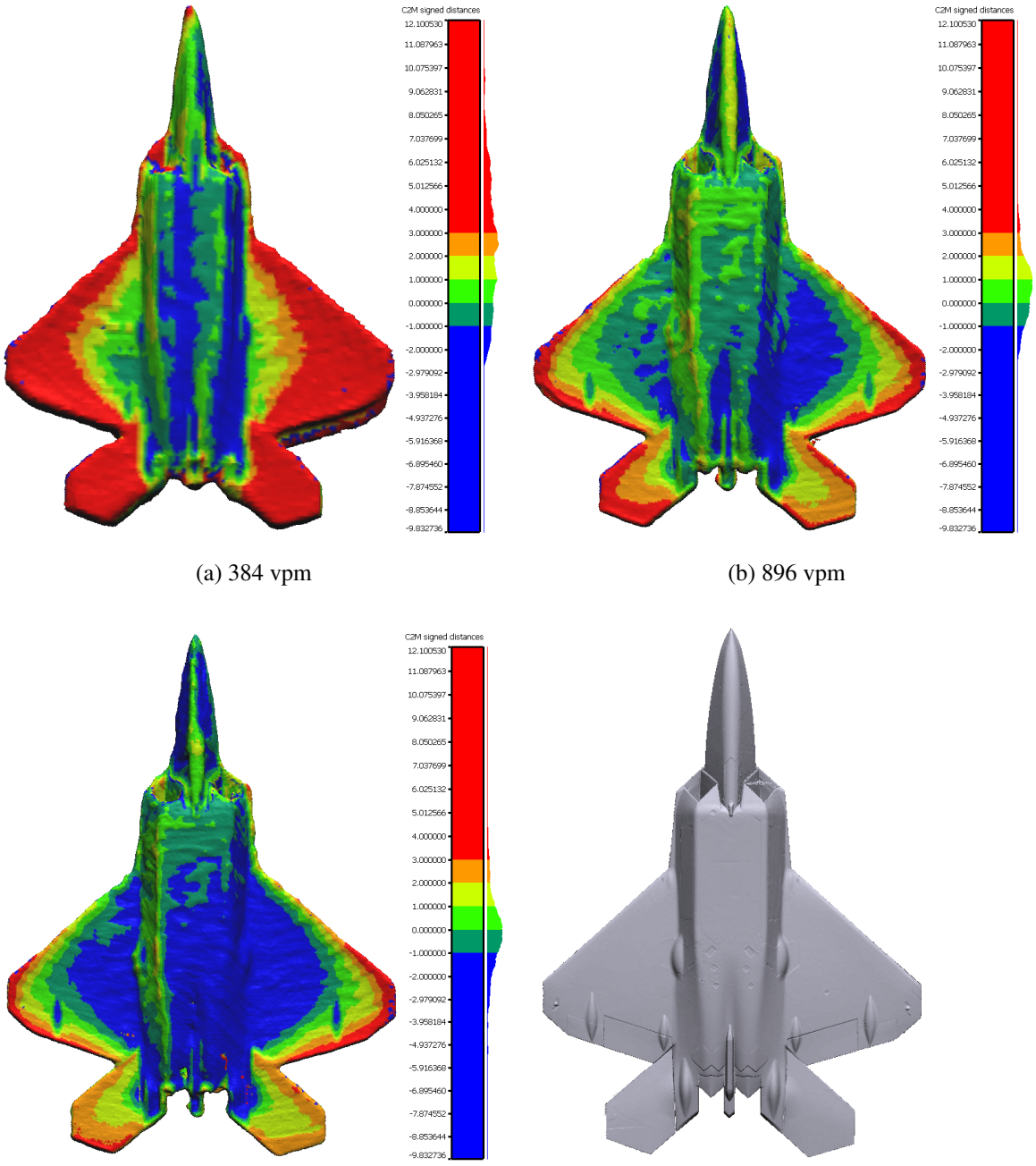
(a) 384 vpm (b) 896 vpm (c) 1280 vpm

Figure 4.4: Visual Comparison of Side View at 0.4 Meters. The above figures provide additional information that can not be readily seen in the top and bottom surface views seen in Figure 4.3. The shading on the vertical tail fins can not be seen well in previous figures, nor can the voxelation that occurs at lower voxel settings. The voxelation is especially noticeable on wings and other thin surfaces in (a). All heat map values and histogram values are in millimeters.

4.1.2 Quality vs Distance.

In Section 4.1.1, the effects of changing the voxels per meter and thereby the resolution of the scan, were explored in detail at three distances. In the previous results, there are differences caused by the change in distance. To isolate the effects caused by distance, this section compares models constructed at the same voxels per meter setting. Specifically, the resolution setting of 896 voxels per meter is used because it is the highest resolution at which a complete model can be built at all three distances.

The results presented in Table 4.4 and Figure 4.6 show a significant decrease in the median model distance error at 0.4 meters (0.388 mm) compared to the median model distance error at 0.8 meters (0.636 mm). Surprisingly, for the models built at the 0.4 meter distance, the IQR is higher than the IQR at 0.6 meters, indicating that there is more



(a) 384 vpm

(b) 896 vpm

(c) 1280 vpm

(d) FaroArm Edge

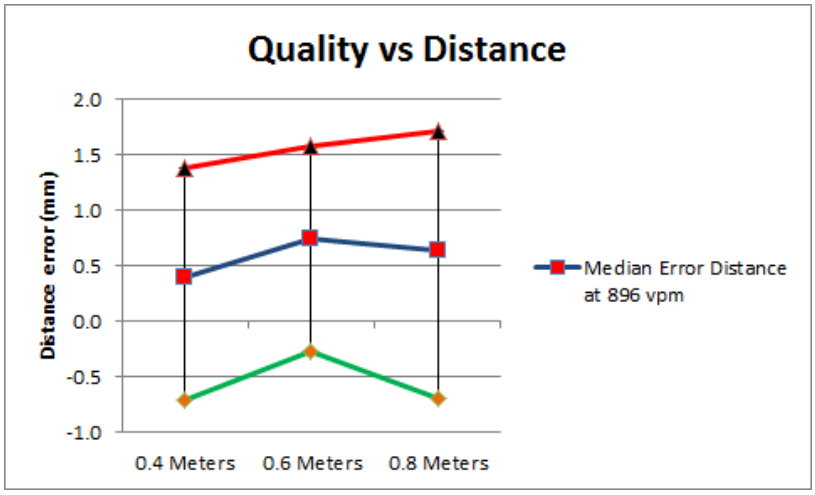
Figure 4.5: Voxelation Effects. The above figures highlight the voxelation effects caused by low voxel per meter settings. In (a), the wings are modeled with large voxels, and are modeled so thickly that there is no indicate that the wings are anything but a smooth surface. As the size of each voxel decreases, the wings are modeled thiner and thiner, which allows the details of the wing to stand out. As seen in (d), there are four deviations on the bottom surface of the wings, which can not be seen at 384 voxels per meter. The larger of the two items can be discerned from the wings when the voxels are increased to 896 per meter (b), and even more clearly at 1280 voxels per (c). All heat map values and histogram values are in millimeters.

Table 4.4: Statistical Analysis of models at varying distances. The below values are the statistical analysis of the combined experimental model building runs for each of the distance settings at the 896 vpm setting.

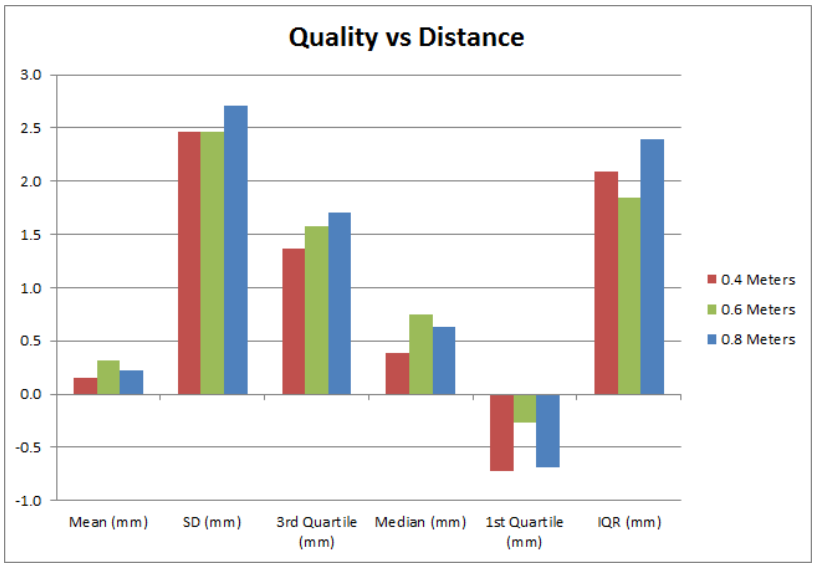
896 Voxels Per Meter	0.4 Meters	0.6 Meters	0.8 Meters
Samples	461100	459219	454807
Mean (mm)	0.2	0.3	0.2
SD (mm)	2.5	2.5	2.7
Maximum (mm)	15.4	13.0	15.1
3rd Quartile (mm)	1.4	1.6	1.7
Median (mm)	0.4	0.7	0.6
1st Quartile (mm)	-0.7	-0.3	-0.7
Minimum (mm)	-16.5	-16.8	-17.3
IQR (mm)	2.1	1.8	2.4

uncertainty in the 0.4 meter results than the 0.6 meter results. The unexpected results highlight the inability for a single value to correctly quantify the quality and accuracy of the models. Just as in Section 4.1.1, the best way to judge the quality is through the visual analysis of the heat map diagrams. The respective heat maps for the three distances at 896 voxels per meter can be seen in Figure 4.7.

With the affects of both vpm and distance on the model quality and accuracy thoroughly analyzed, it is now possible to investigate the absolute detection capabilities of the KinectFusion algorithm.

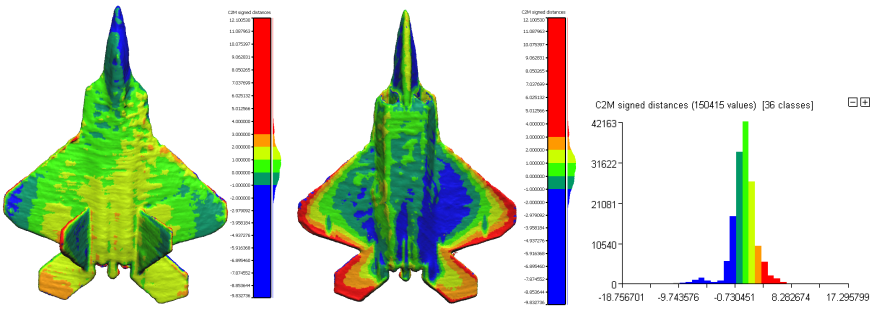


(a) Median Distance Error at 896 voxels per meter

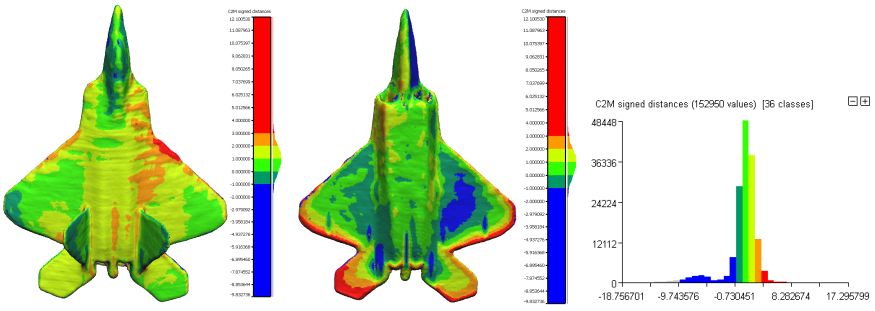


(b) Distance Error Metrics at 896 voxels per meter

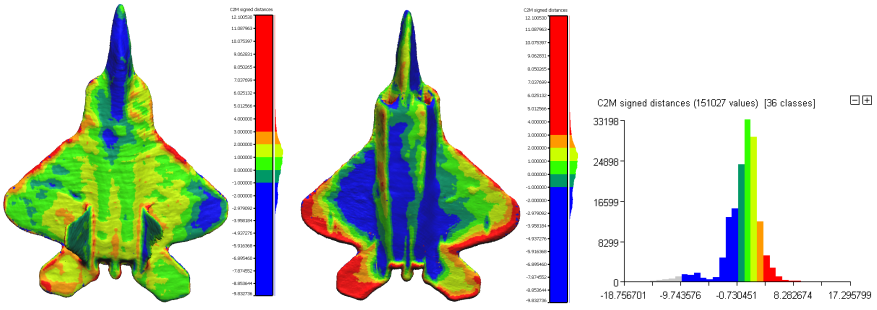
Figure 4.6: Distance Metrics. The change in both accuracy and precision caused by distance can be seen in both (a) and (b). Interestingly, the models built at 0.4 meters is not necessarily better than the model built at 0.6 meters.



(a) 0.4 Meter - Top View (b) 0.4 Meter - Bottom (c) 0.4 Meter - Histogram



(d) 0.6 Meter - Top View (e) 0.6 Meter - Bottom (f) 0.6 Meter - Histogram



(g) 0.8 Meter - Top View (h) 0.8 Meter - Bottom (i) 0.8 Meter - Histogram

Figure 4.7: Visual Comparison of 896 Voxels Per Meter at Various Distances. The affect of increasing the distance by as little as 0.2 meters can be seen in the comparison between (a) and (d). However, it is interesting to note that the edges of the wings appear to have been modeled more accurately at the 0.6 meter distance than at the 0.4 meter distance. All heat map values and histogram values are in millimeters.

4.2 Anomaly Detection Experiment Results

For analysis, the distance values for each coin stack of the four runs were combined and then analyzed. The statistical analysis on the combined distances from the four model runs is seen in Table 4.5.

Table 4.5: Modeled Coin Thickness Analysis. The below values are the statistical analysis of the combined measured thickness of the stacks of coins.

Stack	Samples	Mean (mm)	SD (mm)	Maximum (mm)	3rd Quartile (mm)	Median (mm)	1st Quartile (mm)	Minimum (mm)
1D	520	1.2	0.3	1.7	1.3	1.2	1.0	0.3
1N	520	1.4	0.2	1.8	1.5	1.5	1.3	0.7
1P	520	1.0	0.2	1.4	1.2	1.0	0.8	0.4
1Q	520	1.4	0.2	1.6	1.5	1.4	1.3	0.6
2D	520	2.3	0.3	2.8	2.5	2.4	2.2	1.3
2N	520	3.2	0.3	3.8	3.5	3.2	2.9	2.4
2P	520	2.7	0.2	3.3	2.9	2.8	2.6	1.9
2Q	520	3.1	0.3	3.7	3.3	3.1	3.0	2.1
3D	520	4.0	0.3	4.5	4.1	4.0	3.8	3.2
3N	520	5.5	0.4	6.3	5.9	5.6	5.2	4.2
3P	520	4.2	0.2	4.9	4.4	4.3	4.1	3.4
3Q	520	5.3	0.4	6.3	5.5	5.3	5.0	4.3
4D	520	5.3	0.3	5.9	5.5	5.3	5.0	4.4
4N	520	7.9	0.4	8.6	8.2	8.0	7.6	6.6
4P	520	6.2	0.3	6.9	6.4	6.2	6.0	5.5
4Q	520	7.1	0.4	8.0	7.3	7.1	6.8	6.1
5D	520	6.4	0.4	7.0	6.6	6.4	6.1	5.5
5N	520	9.4	0.6	10.3	9.8	9.4	8.9	7.3
5P	520	7.4	0.3	8.0	7.6	7.3	7.2	6.6
5Q	520	8.8	0.4	9.6	9.2	8.9	8.4	7.5
6D	520	7.7	0.3	8.4	7.9	7.7	7.5	6.9
6N	520	10.4	0.6	11.4	11.0	10.5	10.0	8.7
6P	520	8.5	0.5	9.7	9.0	8.5	8.1	7.3
6Q	520	9.9	0.6	10.7	10.3	9.9	9.6	7.9

Unfortunately, it was discovered that while the diameter of US coins are tightly controlled, the thickness is not. Due to variations in the striking force applied when a coin is struck, the thickness of the coin will vary, and therefore the thickness is not as tightly controlled as the diameter. Given the inconsistencies in the coin thickness, the height of each stack of coins were measured using a set of calipers with an error of ±0.02 mm for comparison to the modeled thicknesses. The comparison of the modeled coin stack heights and the true measurement of the coin stacks can be seen in Table 4.6.

The results of the coin study indicates that the KinectFusion based models are significantly too small in the case of a single coin. The worst case was the single penny, with an actual thickness of 1.45 mm and a median modeled thickness of 0.959 mm, a difference of 33.39%. The best case was a bit surprising, as a dime, which presents the smallest surface area to the sensor for modeling, yielded the closest match at a difference of only 9.82%, which is however, still one the largest differences of the 24 coin stacks.

As expected, as the true height of the coin stack increases the difference between the true height and modeled height decreases. Upon detailed review however, there is an interesting trend in the difference of modeled height. Starting at a single coin, in the range of 1.30 mm to 1.83 mm, the modeled height is below the actual height, and the difference begins to decrease as stack height increases. In the 4 mm to 6 mm range, the difference in heights continues to decrease and the modeled height even becomes taller than the actual coin height. However, above the 6 mm range, there is a reversal again, in that the difference increases in so much that the modeled is once again shorter than the actual height. This trend can be seen in Figure 4.8. This chart is ordered with the smallest true thickness on the left and the largest on the right, similar to the coin arrangement on the flat surface. A possible cause of this trend is discussed later.

Table 4.6: Actual vs Modeled Coin Thickness Values. The below values are the mean and median thickness of the modeled stacks of coins compared to the true measured thickness.

Stack	Mean (mm)	Median (mm)	True Thickness (mm)	Difference	% Difference
1D	1.2	1.2	1.3	0.1	9.8%
1N	1.4	1.5	1.8	0.4	22.4%
1P	1.0	1.0	1.5	0.5	33.4%
1Q	1.4	1.4	1.7	0.3	19.9%
2D	2.3	2.4	2.5	0.2	6.7%
2N	3.2	3.2	3.6	0.4	12.1%
2P	2.7	2.8	2.9	0.2	6.1%
2Q	3.1	3.1	3.4	0.3	7.7%
3D	4.0	4.0	3.9	0.0	-0.5%
3N	5.5	5.6	5.6	0.1	1.2%
3P	4.2	4.3	4.5	0.2	5.2%
3Q	5.3	5.3	5.2	-0.1	-1.7%
4D	5.3	5.3	5.2	-0.1	-1.3%
4N	7.9	8.0	7.7	-0.2	-2.2%
4P	6.2	6.2	5.9	-0.3	-5.2%
4Q	7.1	7.1	7.0	-0.1	-1.1%
5D	6.4	6.4	6.6	0.2	3.0%
5N	9.4	9.4	9.3	-0.1	-0.8%
5P	7.4	7.3	7.2	-0.2	-2.1%
5Q	8.8	8.9	8.6	-0.2	-2.4%
6D	7.7	7.7	7.7	0.1	0.8%
6N	10.4	10.5	11.4	0.9	8.1%
6P	8.5	8.5	8.7	0.1	1.5%
6Q	9.9	9.9	10.3	0.4	3.9%

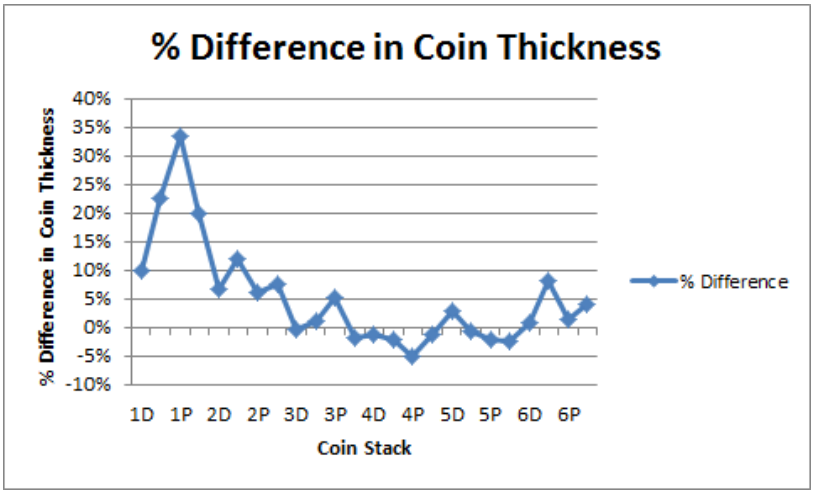


Figure 4.8: Percent Difference in Actual Height vs Modeled Height. This chart highlights the trend in the difference between the actual coin stack height vs the modeled coin stack height. Differences are presented as a percentage of the true height. Of particular interest is the trend from the smallest coin stacks to the largest. The decrese in error percentage followed by an increase is caused by the radial distortion inherent to the Kinect sensor, as discussed by Khoshelham and Elberink [20].

V. Conclusion

This work provides the first quantifications of model quality for the KinectFusion algorithm and the Kinect sensor. The main result here is that the KinectFusion algorithm provides high quality measurements based on noisy, low resolution sensor data. Recall that the Kinect sensor only provides depth imagery with VGA resolution (640x480 pixels) per frame and has an error of approximately ±10 mm [20]. When paired with the KinectFusion algorithm, the combined system is able to produce models with approximately ±3 mm of error, a dramatic improvement over the sensor alone.

Additionally, while discerning a single dime on a flat surface is an easy task for the human eye and brain, it is no simple task for a low resolution sensor, even at 0.4 meters away. Nevertheless, by stitching multiple depth frames together from varying perspectives, the resulting model can detect a surface deviation as little as 1.30 mm in thickness. Although the system is inaccurate (off by as much as 33%) with regard to the absolute dimensions of the surface deviation caused by a 1.30 mm object, the ability to detect such a deviation with a noisy sensor is impressive. Along with the impressive overall capabilities of the KinectFusion, there are some interesting observations that warrant a closer look.

In both the F-22 and coin experiments there appears to be evidence of a sensor bias. For the coins experiment, the trend of decreasing height differences followed by increasing height differences (Figure 4.8) highlights that there are clear limitations when modeling near the edges of the sensor's field of view vs the center of the sensor's field of view. In the coin experiment, the single coin of each denomination was placed on the far left side of the experiment, while the six coin stack of each denomination was place on the far right side. The camera was then moved from the front of the surface to above the surface looking down, as if following a 0.4 meter sphere centered on the center of the coins, approximately between the stacks of three pennies and quarters and the stacks of four. Given that the

differences in height for stacks of the three and four quantities are lower in the center of the sensors view compared to the higher differences at the edges of the sensor, this is definitely evidence of a possible sensor bias. This is likely caused by the radial distortion of the Kinect sensor, as noted and quantified by Khoshelham and Elberink [20].

Along with the difference trend first discussed in Section 4.2, there is also a possible bias in the model based on where the sensor started. Figure 5.1 shows a slope of the top surface of the coin stacks. The lowest part of the slope corresponds to the front edge of the coin stack, and the location at of the sensor when the model building began. As described earlier, the sensor started at the front at approximately a 30° angle to the flat surface, and was then rotated over the flat surface approximately 150° from the starting location. This sloping indicates a possible bias in the KinectFusion algorithm based on the sensor's starting position.

Similarly, this starting position bias is also seen in the model accuracy experiment in the horizontal stabilizers of the model aircraft. An inspection of the heat map representations (Figure 4.3) reveals a stark contrast in the distance error values on the top surface of the horizontal stabilizers as compared to the distance error values on the bottom surface of the horizontal stabilizers. This may be attributable the experiment setup (Section 3.1.1), wherein the model aircraft was suspended at approximately a 43° nose up angle to ensure the sensor would be able to see both the top and bottom surfaces of the airplane during a 360° rotation. The scanning session began at the rear of the aircraft model, with the top surface of the horizontal stabilizers at approximately 0.4 meters (or 0.6 or 0.8 meters depending on operating distance) away from the sensor. Once the sensor rotated 180° around the model, the bottom side of the horizontal stabilizers were approximately 0.4meters + 0.28meters = 0.68meters away from the sensor, where 0.28 meters is the length of the aircraft when at a 43°pitch angle.

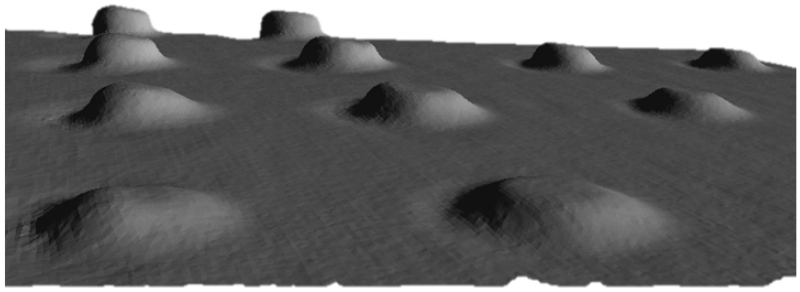


Figure 5.1: Possible Sensor and Algorithm Bias. This figure highlights a possible bias of the sensor and algorithm. Note the sloped top surface of the modeled coin stacks, where the low point of the slope is the direction of the initial sensor position.

5.1 Future Work

This section presents several areas and ideas that build and expand on the work presented by this experiment.

5.1.1 Programmatic Analysis of Coin Dimensions.

The two tools utilized for point cloud and mesh analysis in this work, CloudCompare and MeshLab, offer tremendous capabilities, of which this work utilized an extremely small percentage of. However, when it came to the analysis of the coins, the best analysis technique that could be used was the manual picking of points on the top surface of a coin stack.

There is significant room for a more detailed analysis on the coin stack point clouds and models. The first four points picked define a plane for the specific stack of coins, but it also defines a bounding box as well. Any points on the stack must be within this bounding box, and this can be used to quickly eliminate a vast majority of points from the entire point cloud, which drastically reduces the number of computations that must be performed and thus the time required for analysis on each coin stack.

From there, the normal vector at each vertex, based on calculations in MeshLab, can be compared to the normal vector of the planar surface used as the zero height surface in

the analysis for Section 4.2. In order to sample all the points on the top surface of the stack of coins, as opposed to the relatively few 30 points that were picked manually for this work, any point with a normal vector parallel to the planar normal vector within a specified margin of error could be used. This would increase the sample size of the experiment, and help improve the accuracy of the results. Similar techniques could be applied to measure the diameter of the coin stacks, which was simply too imprecise to do with manual point picking.

5.1.2 KinectFusion and Different Sensor Technologies.

Given the KinectFusion's tremendous improvements in model quality when paired with a Kinect sensor, it will be interesting to see if the same level of improvement would be obtained when pairing the KinectFusion algorithm with a higher fidelity sensor, such as a Light Distance and Ranging (LiDaR) or the newly released Xbox One Kinect sensor. Such devices provide higher resolutions than the Kinect alone, thus providing much less noisy data to the KinectFusion algorithm. Would a two-fold or three-fold improvement still be possible with LiDaR or Xbox One data?

5.1.3 KinectFusion and Vicon Info.

Another source of noise (or error) in the KinectFusion algorithm is the pose estimation calculations. Section 2.3.3 discussed how the KinectFusion algorithm makes use of two different methods of pose estimation, which is vital to the success of the model building process. During experimental runs, the model quality would deteriorate or completely dissolve the model if the pose estimation algorithm were unable to calculate the estimated pose of the sensor. The two currently implemented methods for pose estimation rely on the noisy model or the noisy sensor. One future alternative might be to provide higher fidelity pose information to the KinectFusion algorithm, via an external reference system like the Vicon motion capture system. Vicon systems rely on an array of high speed IR cameras in a

calibrated configuration and can yield position information in the sub-millimeter accuracy range.

The first step would be to perform a comparison of the estimated pose determined by the KinectFusion algorithm and the actual pose location as given by the Vicon system. From there, the KinectFusion algorithm and application could be modified to make use of the highly accurate Vicon system. The system would likely need to use the Vicon system as a boot strap that maintains pose estimation even when model-based tracking fails, rather than replacing the pose estimation entirely.

5.1.4 KinectFusion and Feature Detection.

The current configuration of the KinectFusion algorithm and the pose estimation methods are designed to work without any preexisting knowledge of the object being scanned. However, there are use cases where a user is not scanning an unknown object, but rather is scanning an object for which they have a CAD model. Due to this existing knowledge, the KinectFusion algorithm could be modified to include feature detection algorithms, such as scale-invariant feature transform or spin image feature detection, to improve the capabilities of its pose estimation and ultimately the resulting model.

For instance, in the F-22 experiment, the pose estimation techniques struggled when the aircraft was presented at the side angle, i.e. looking down the wings. Since the wings are so narrow, it's like looking down on the edge of a knife. There are very few (if any) depth returns on the wings, leaving just the body to produce depth values. Unfortunately, at this angle, the surface of the body exposed is also at a minimum, which ultimately leads to a very small section of the aircraft for the system to be able to model and align with previous frames. If the system had preexisting knowledge of the F-22, it may be possible to avoid some of the pose estimation tracking loses caused by these low profile areas.

5.1.5 Further Investigation of the Sensor and Algorithm Biases.

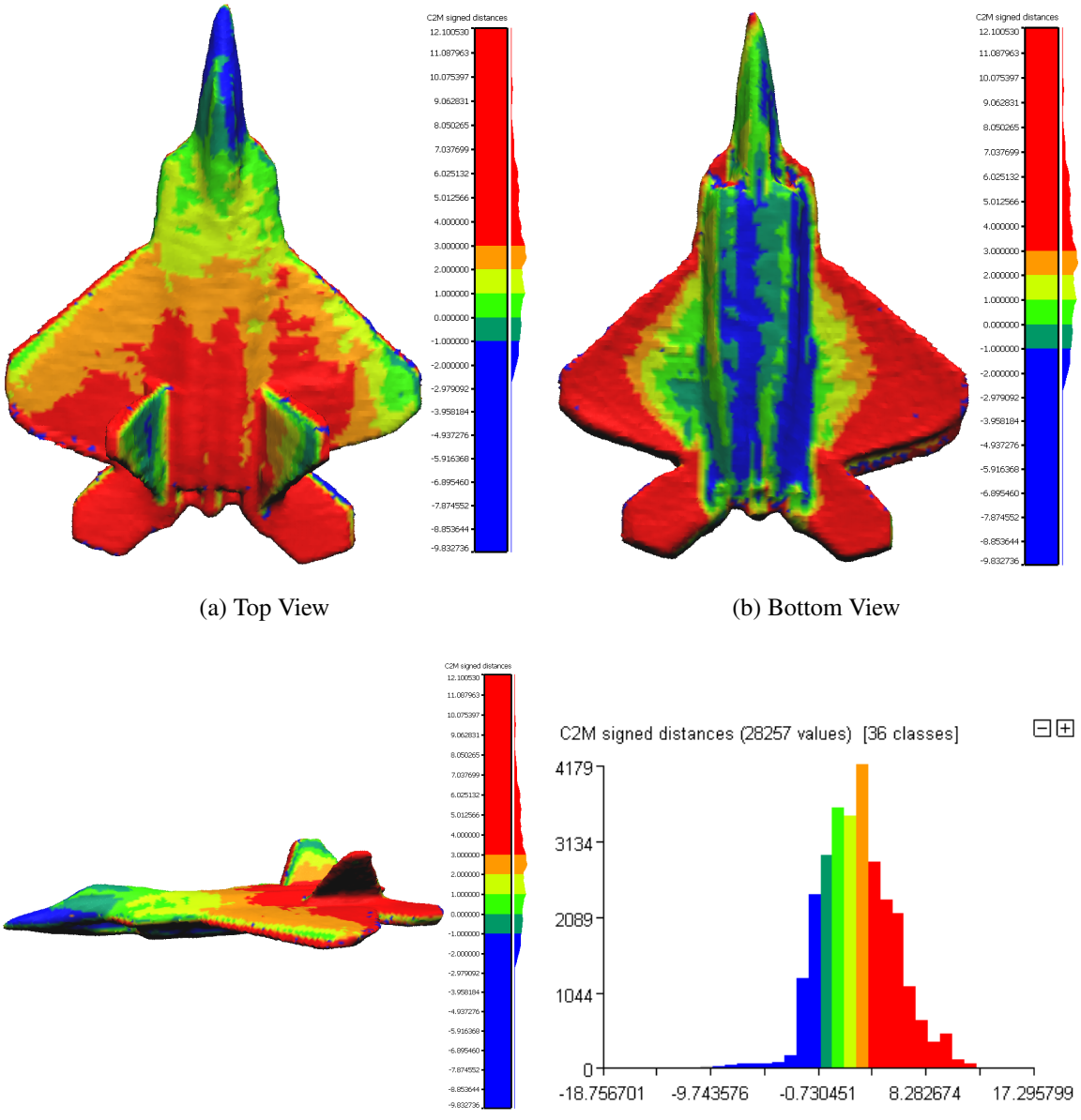
As described earlier, the results point to possible biases in the sensor and the KinectFusion algorithm. In particular, the coin experiment especially highlights a possible bias for objects on the outer edges of the viewing area compared to objects near the center of the viewing area. In order to verify and possibly quantify this bias, the coin experiment should be conducted using different arrangements of the coins (shortest coin stacks in the middle, largest in the middle, etc), as well as starting from different view points with the sensor and with a calibration correction for radial distortion.

5.2 Conclusion

This work provided a quantification of the KinectFusion's ability to improve the quality of a model constructed with the Kinect sensor. The KinectFusion provides a significant improvement in the accuracy of the depth measurements that are obtained from a noisy commodity sensor such as the Kinect. This work opens the doors not only for further analysis, but also the implementation of the Kinect sensor and KinectFusion algorithm in tasks that require millimeter level accuracy and precision.

Appendix: Model Accuracy Heat Maps

This appendix includes all of the generated heat maps. All values are in millimeters.



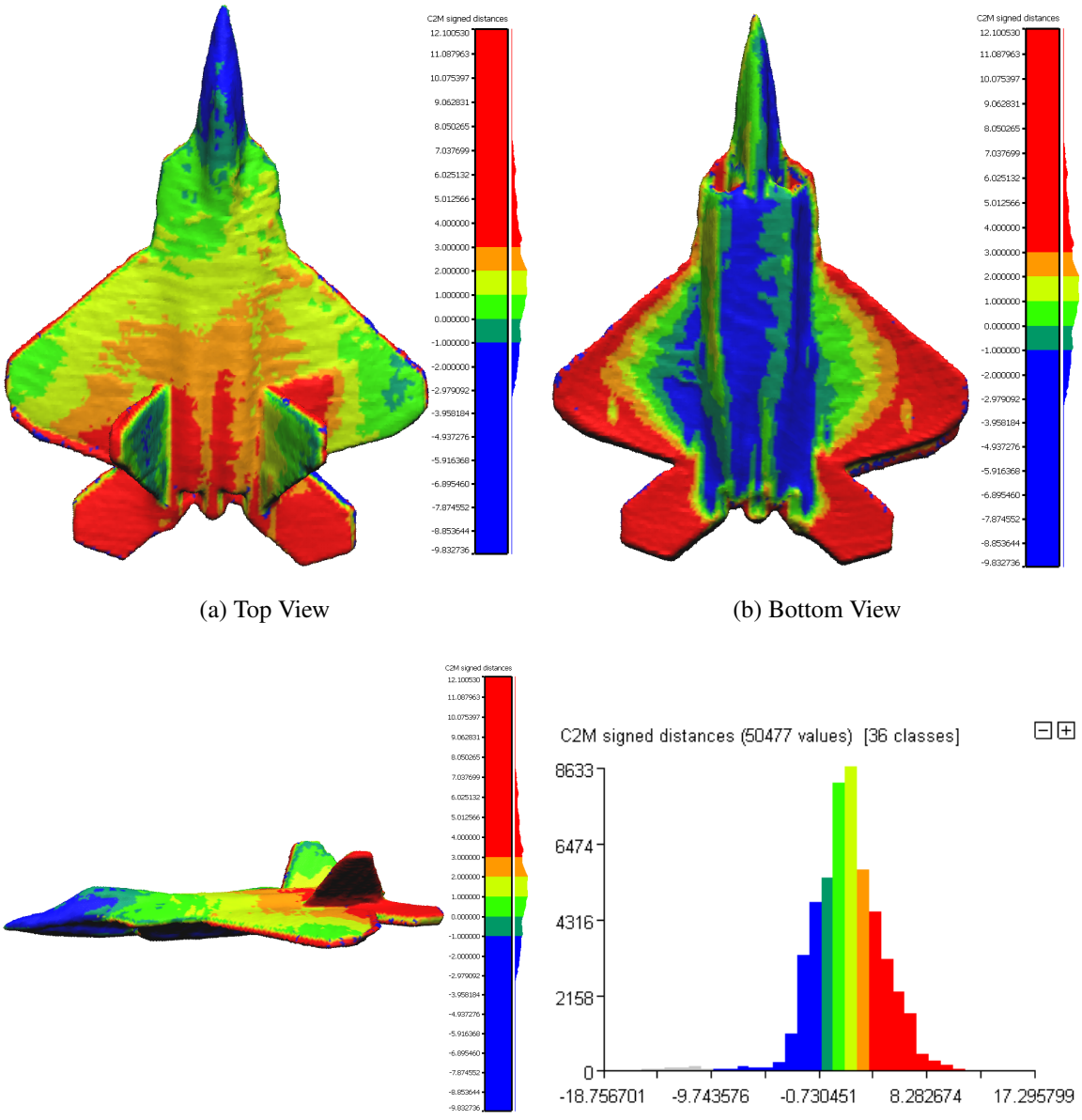
(a) Top View

(b) Bottom View

(c) Side View

(d) Histogram

Figure A.1: 384 vpm at 0.4 Meters.



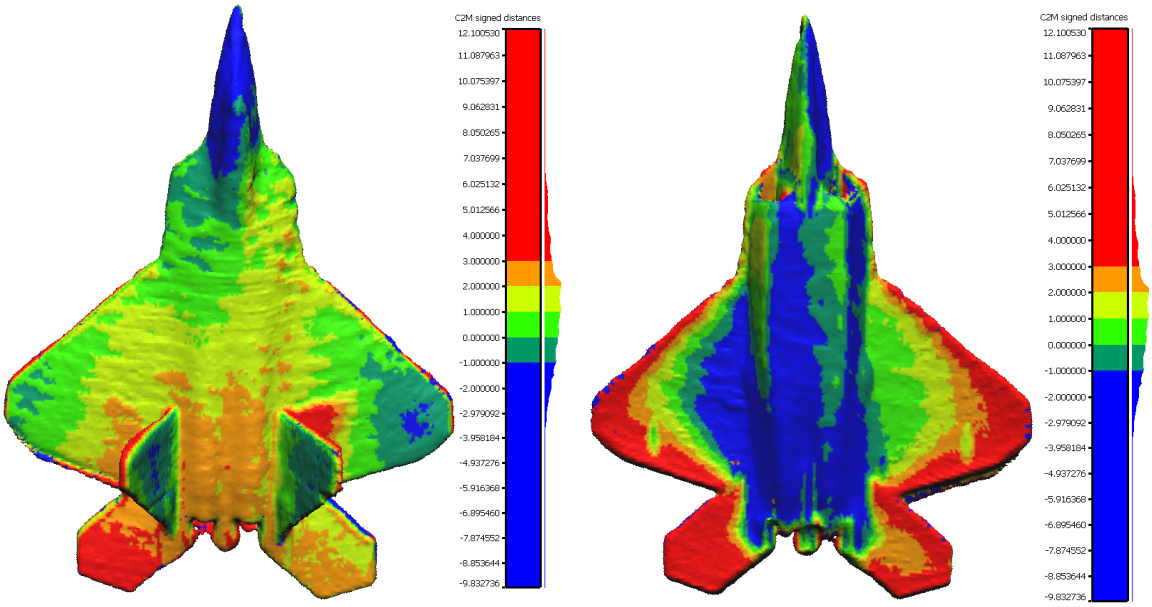
(a) Top View

(b) Bottom View

(c) Side View

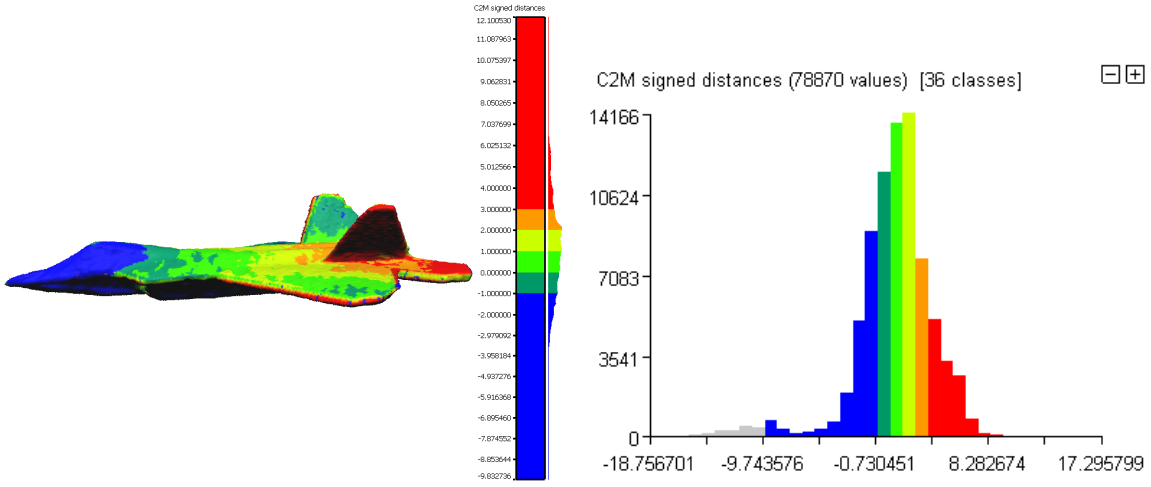
(d) Histogram

Figure A.2: 512 vpm at 0.4 Meters.



(a) Top View

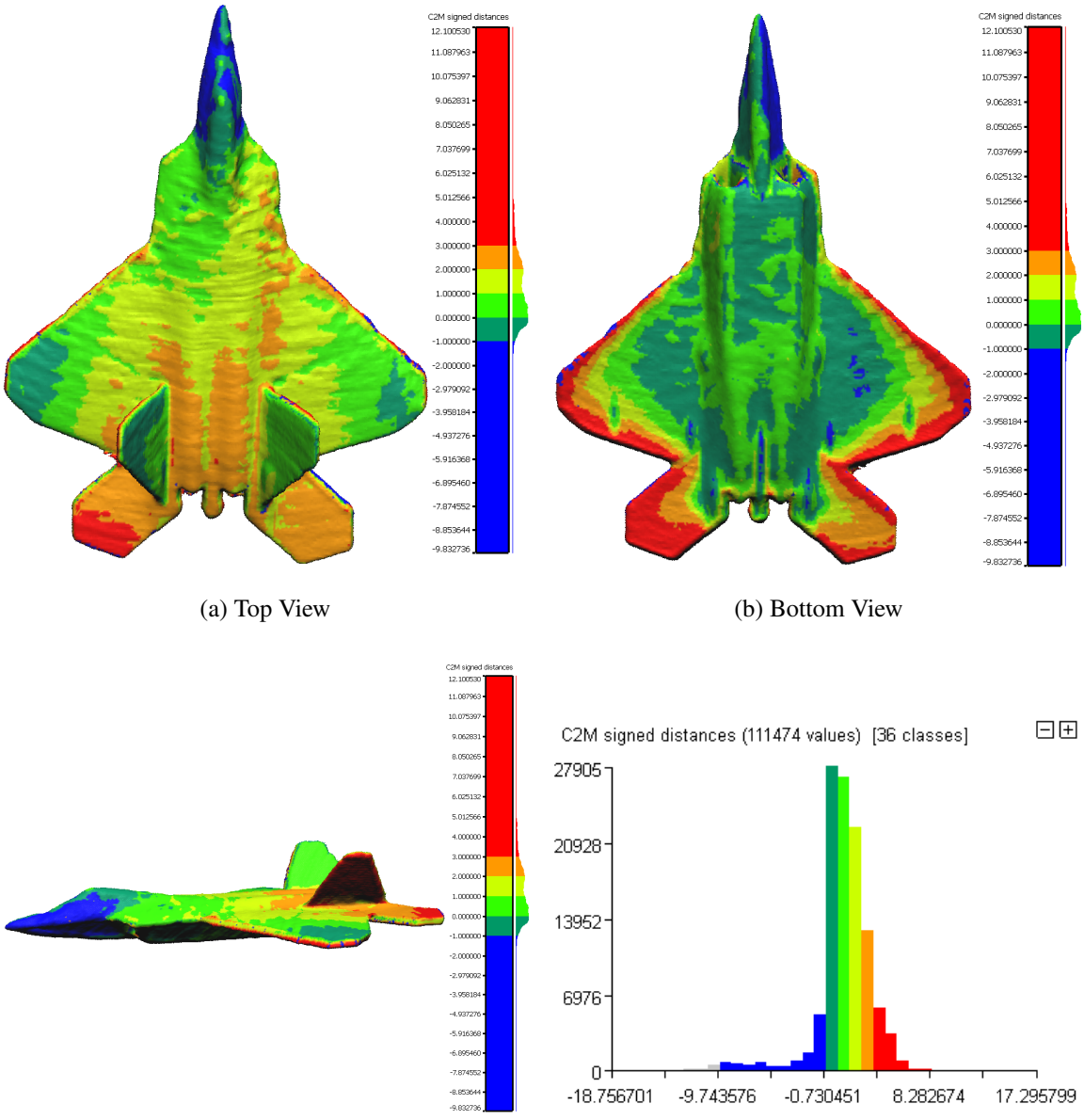
(b) Bottom View



(c) Side View

(d) Histogram

Figure A.3: 640 vpm at 0.4 Meters.



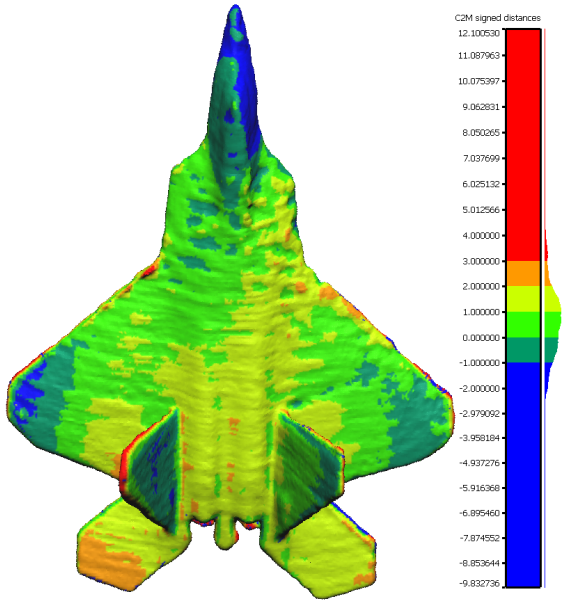
(a) Top View

(b) Bottom View

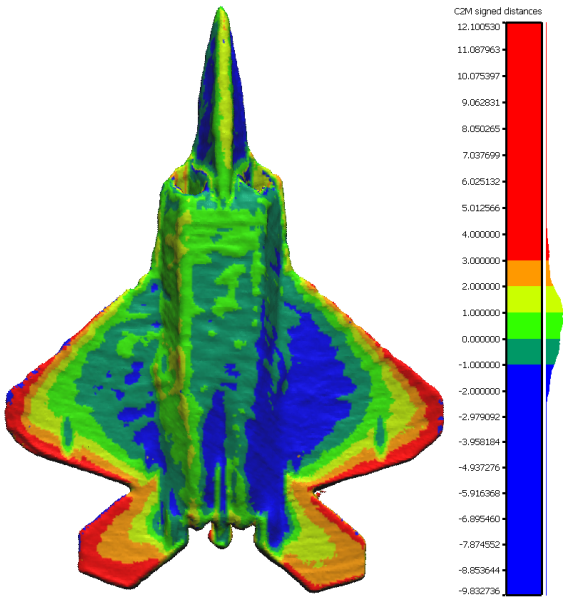
(c) Side View

(d) Histogram

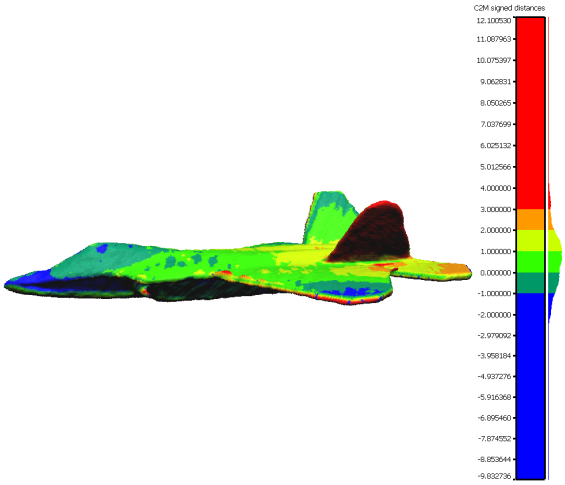
Figure A.4: 768 vpm at 0.4 Meters.



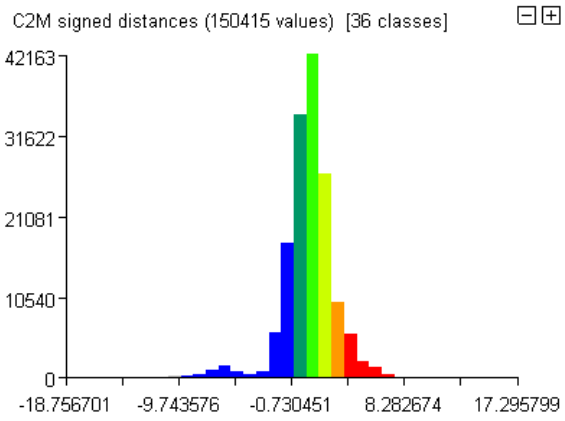
(a) Top View



(b) Bottom View

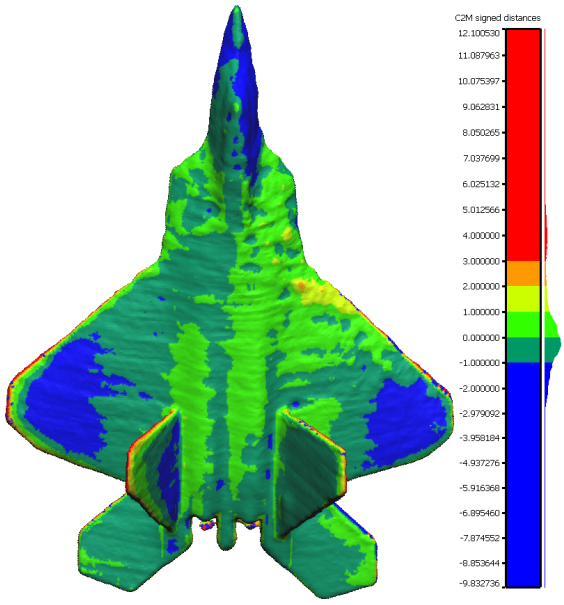


(c) Side View

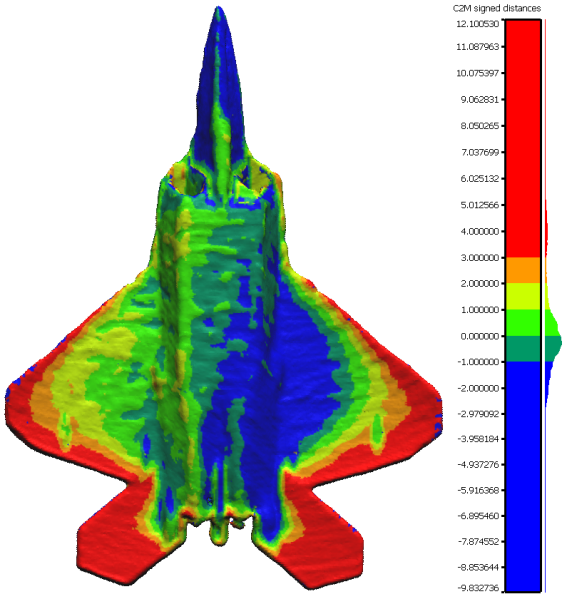


(d) Histogram

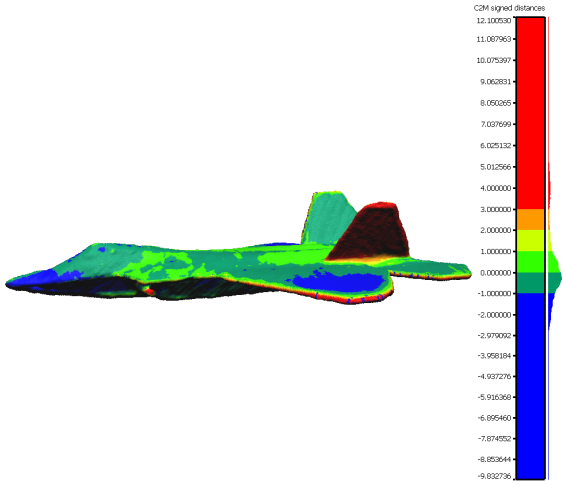
Figure A.5: 896 vpm at 0.4 Meters.



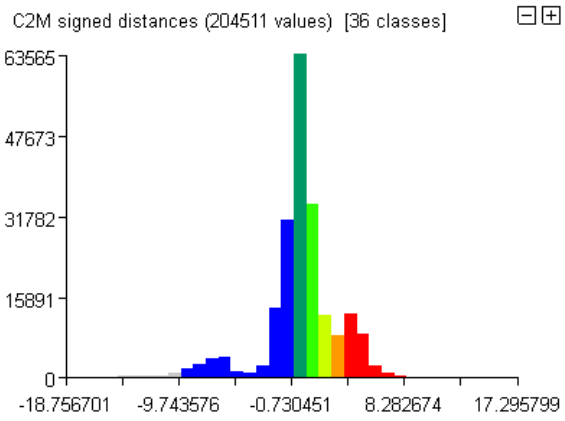
(a) Top View



(b) Bottom View

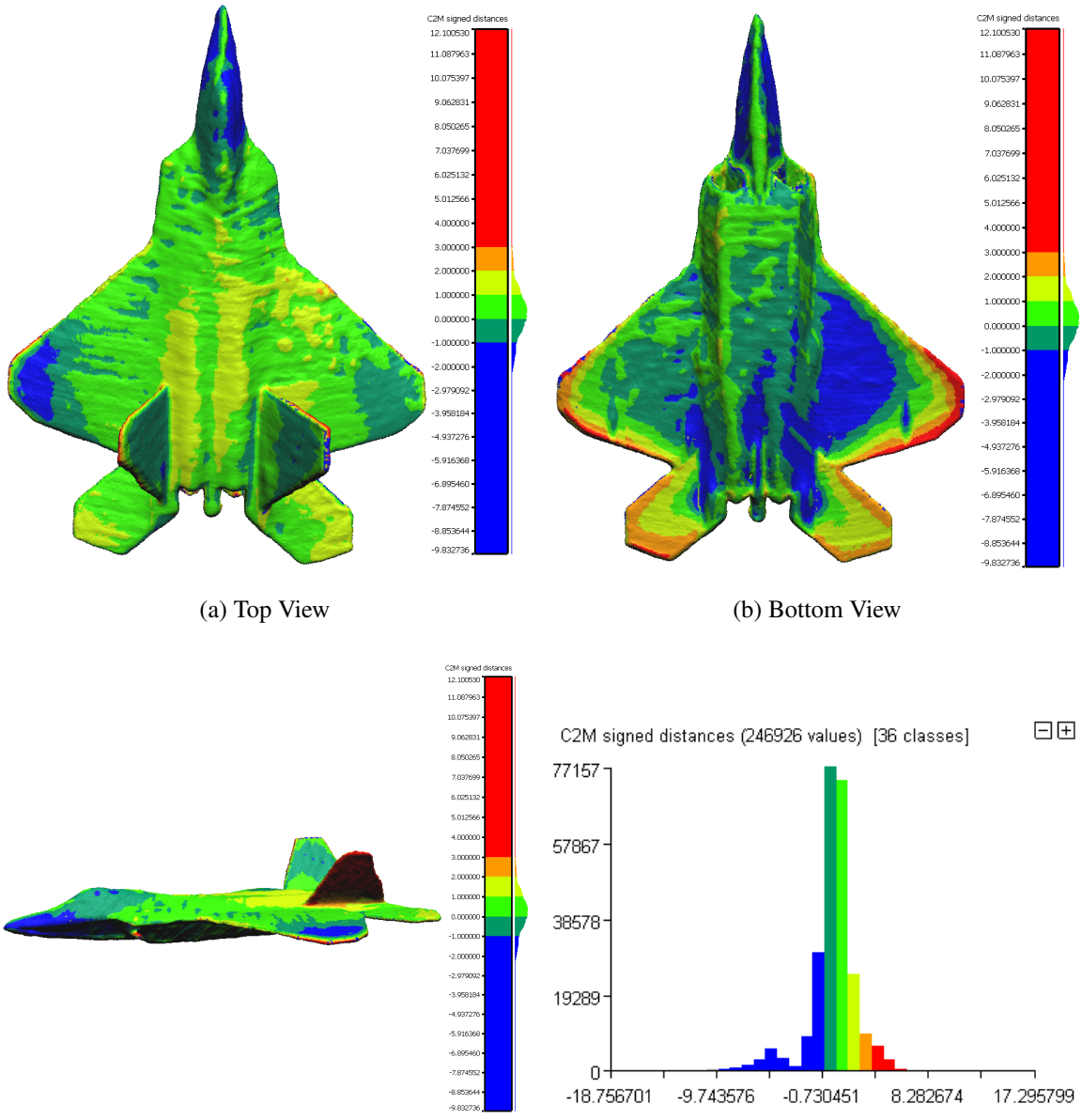


(c) Side View



(d) Histogram

Figure A.6: 1024 vpm at 0.4 Meters.



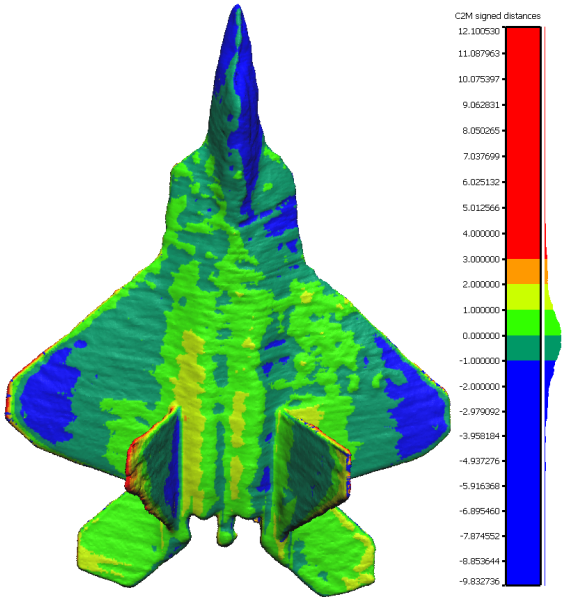
(a) Top View

(b) Bottom View

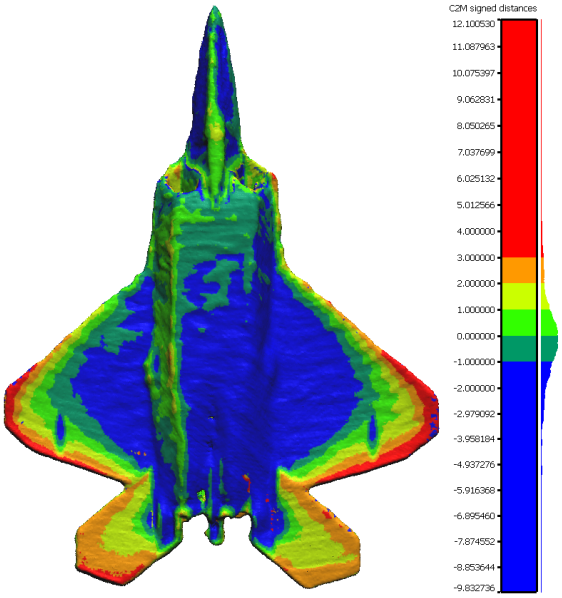
(c) Side View

(d) Histogram

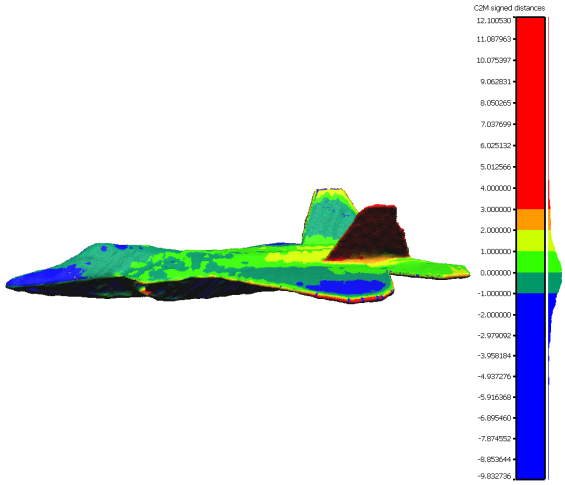
Figure A.7: 1152 vpm at 0.4 Meters.



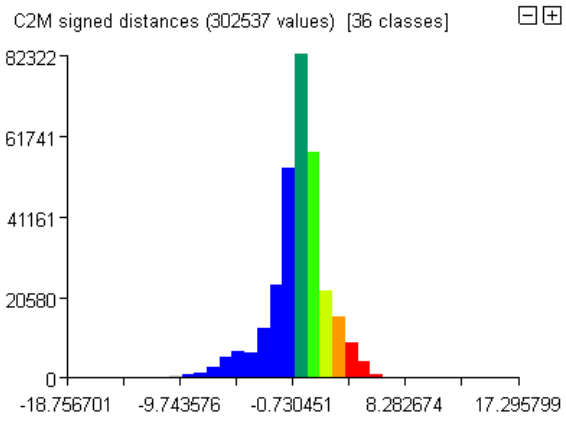
(a) Top View



(b) Bottom View

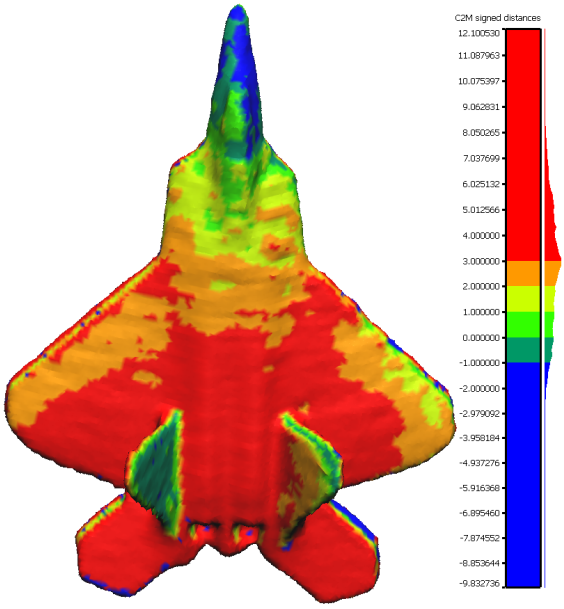


(c) Side View

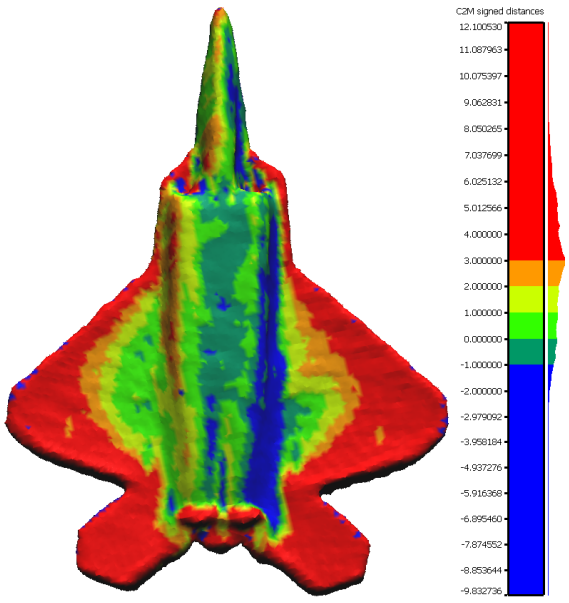


(d) Histogram

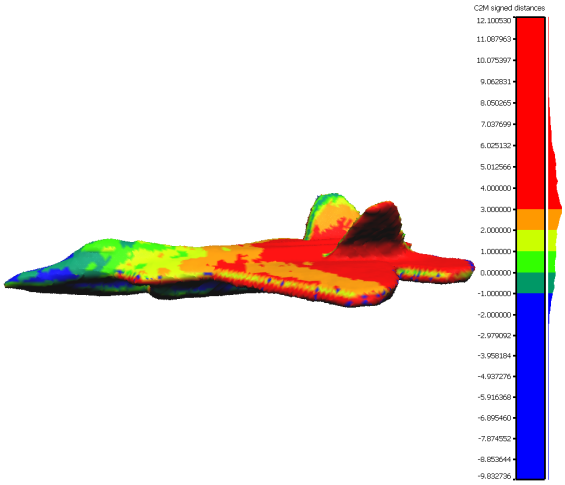
Figure A.8: 1280 vpm at 0.4 Meters.



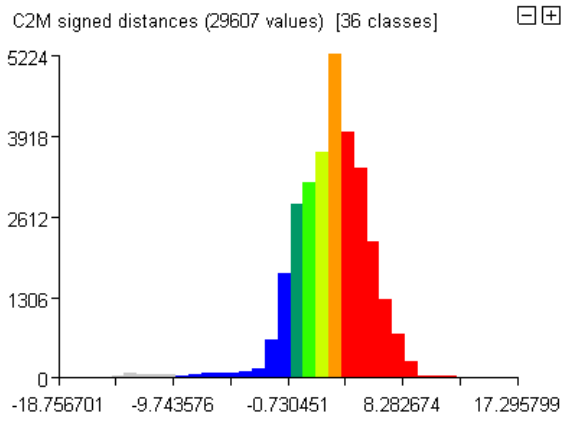
(a) Top View



(b) Bottom View

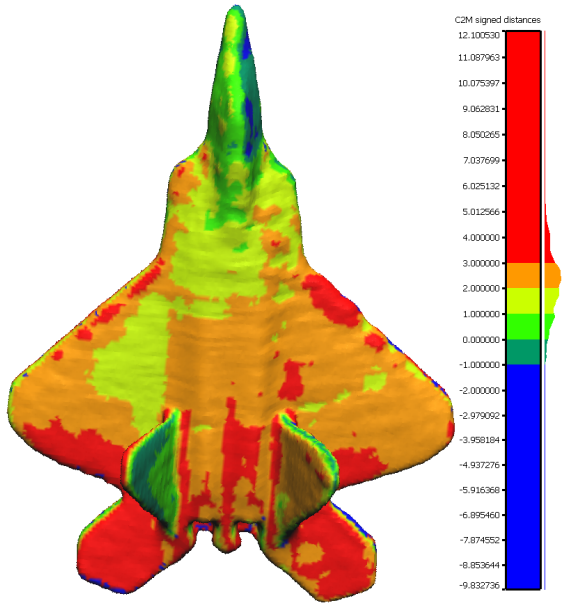


(c) Side View

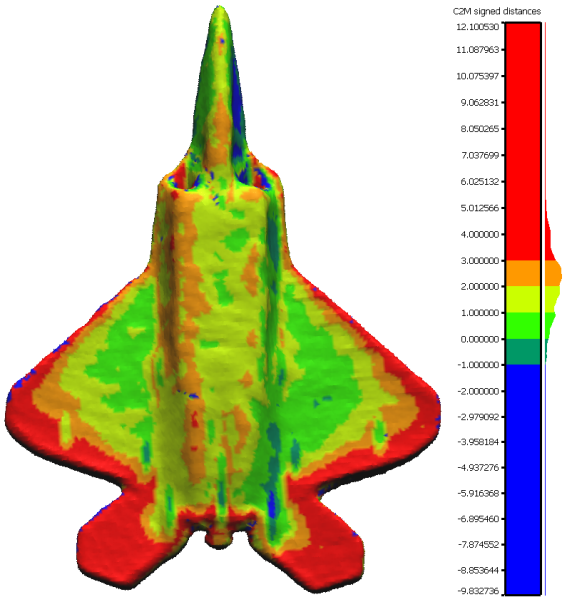


(d) Histogram

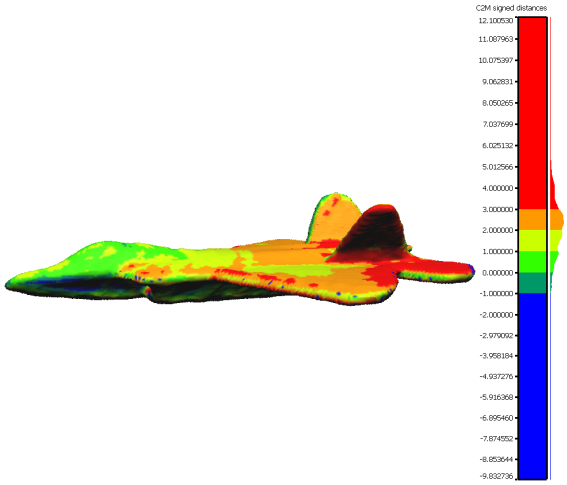
Figure A.9: 384 vpm at 0.6 Meters.



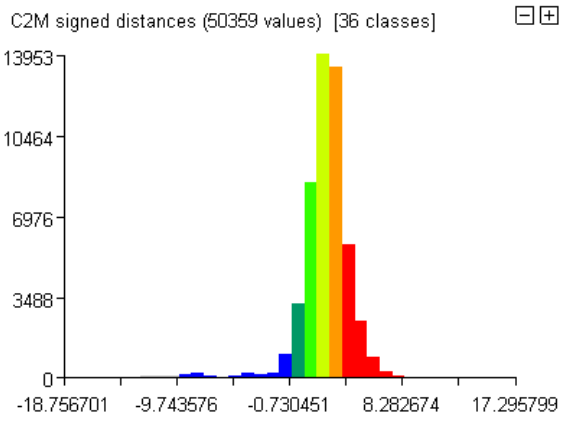
(a) Top View



(b) Bottom View

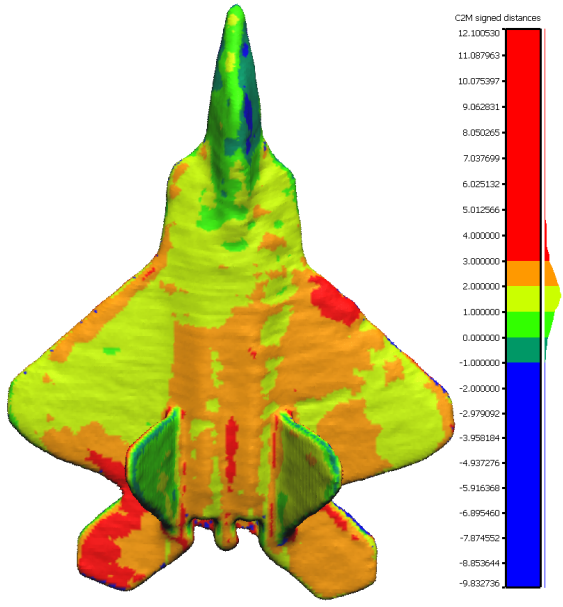


(c) Side View

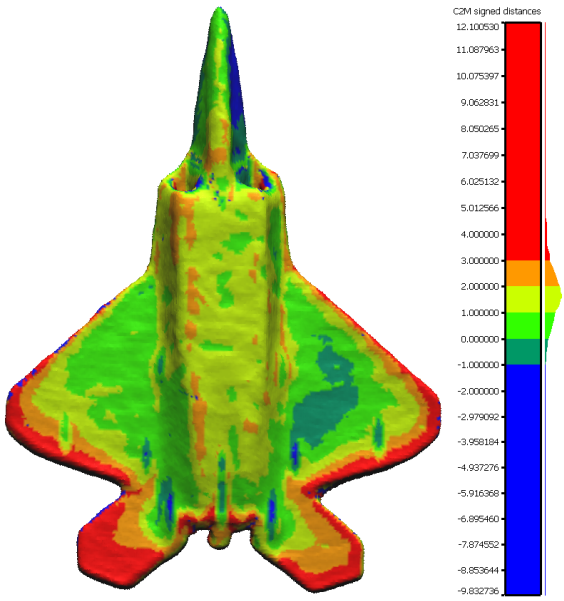


(d) Histogram

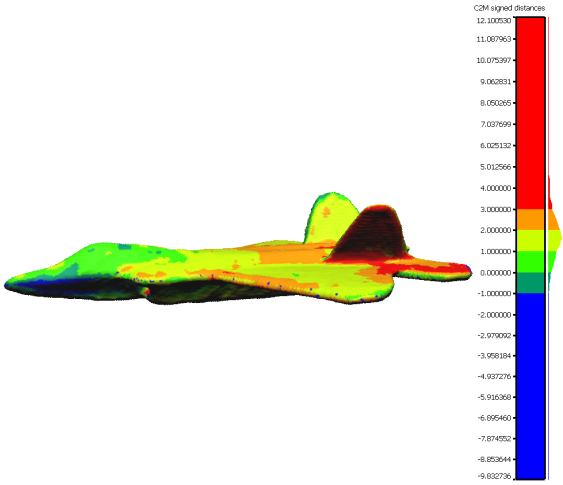
Figure A.10: 512 vpm at 0.6 Meters.



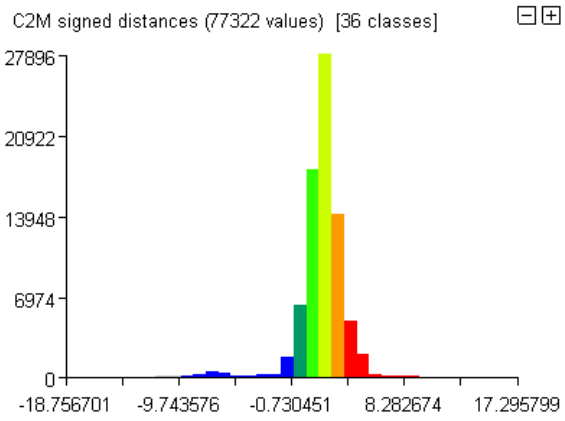
(a) Top View



(b) Bottom View

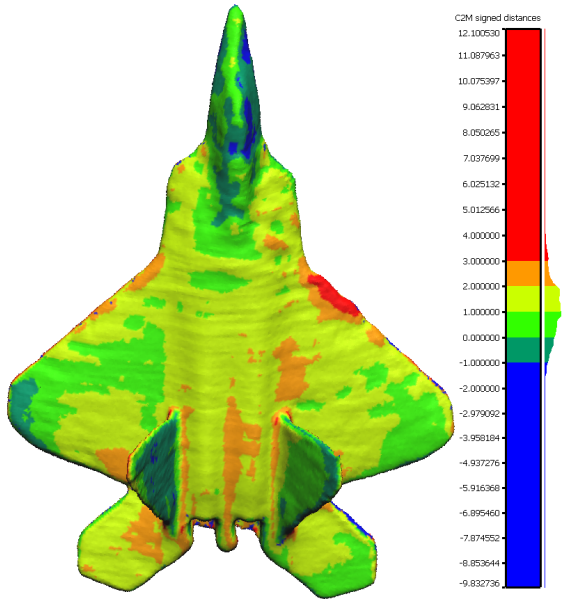


(c) Side View

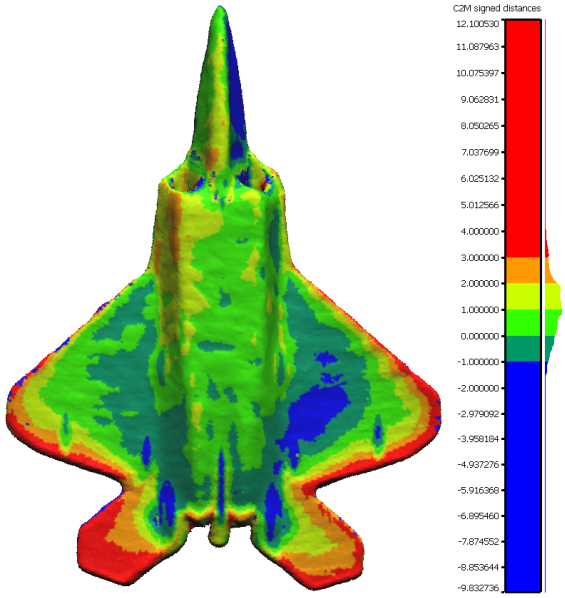


(d) Histogram

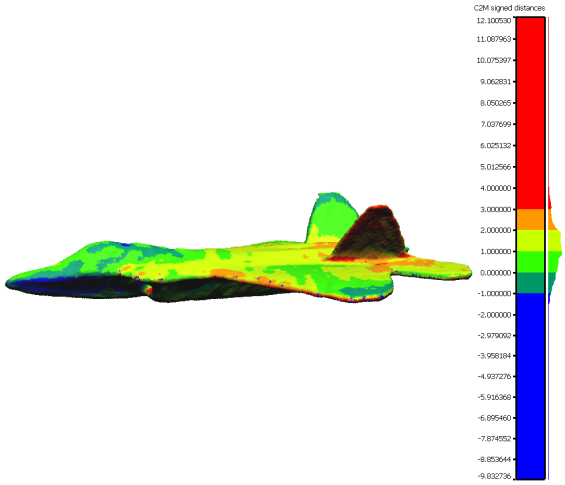
Figure A.11: 640 vpm at 0.6 Meters.



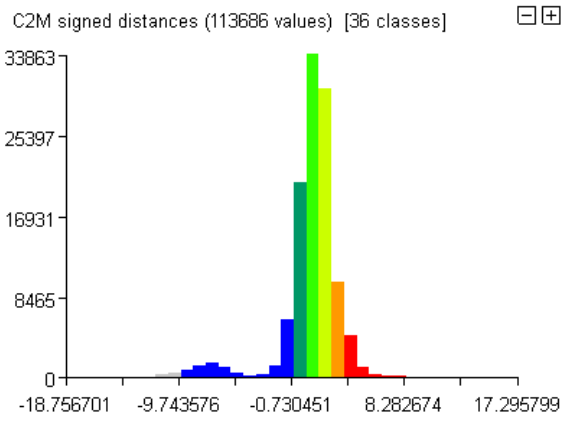
(a) Top View



(b) Bottom View

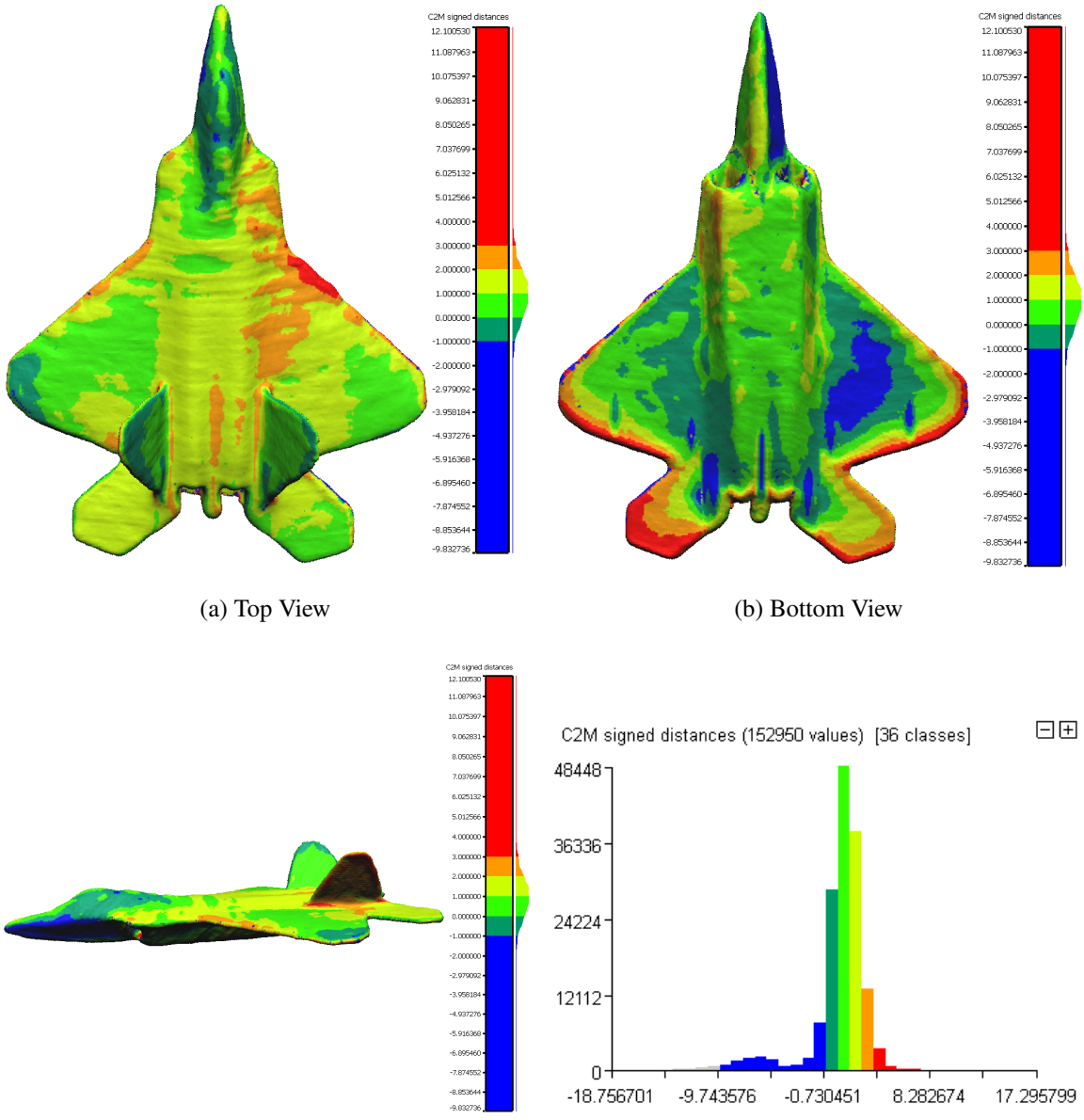


(c) Side View



(d) Histogram

Figure A.12: 768 vpm at 0.6 Meters.



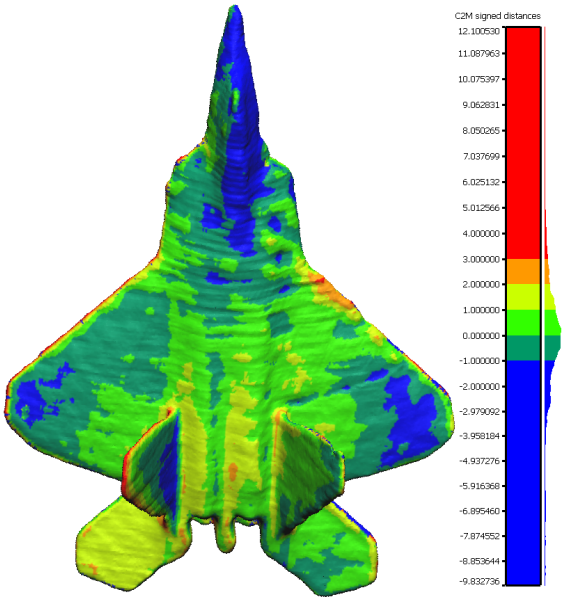
(a) Top View

(b) Bottom View

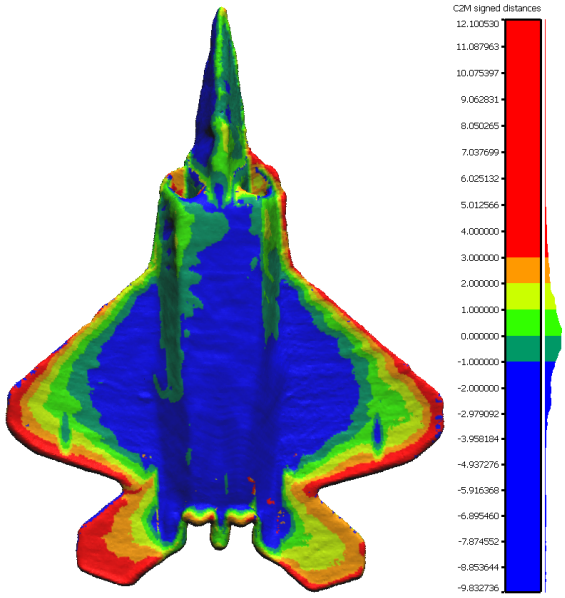
(c) Side View

(d) Histogram

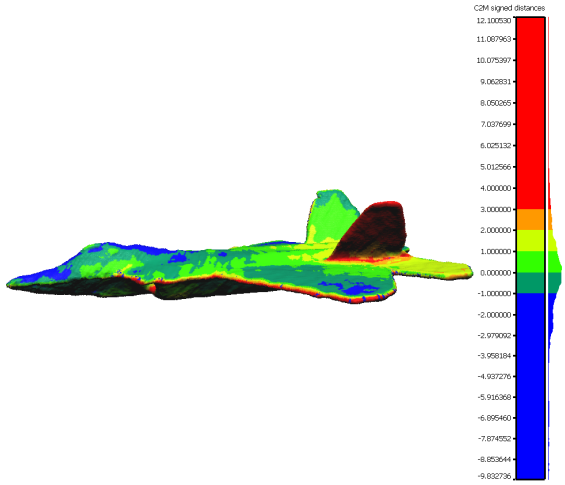
Figure A.13: 896 vpm at 0.6 Meters.



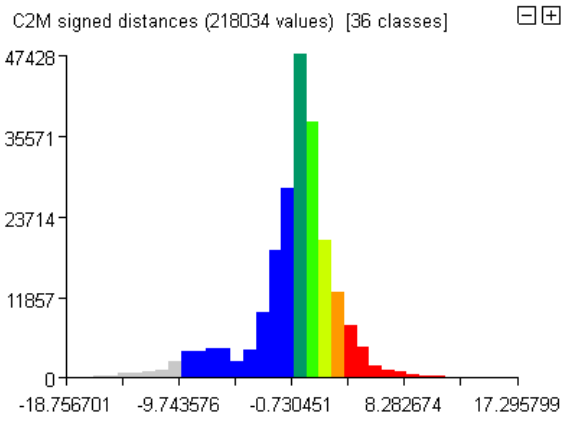
(a) Top View



(b) Bottom View

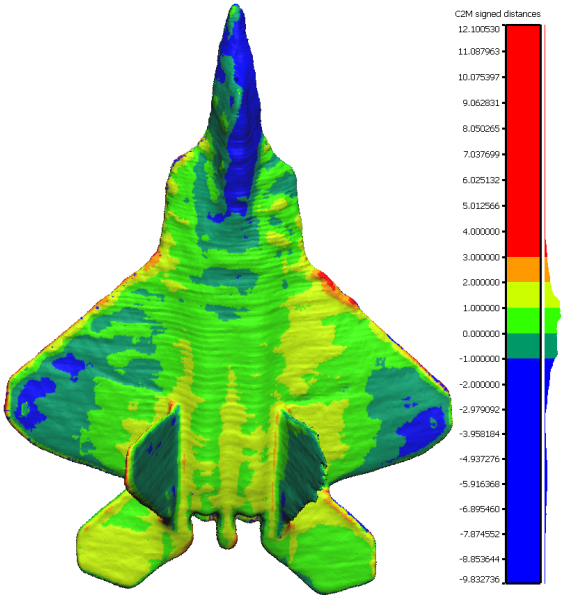


(c) Side View

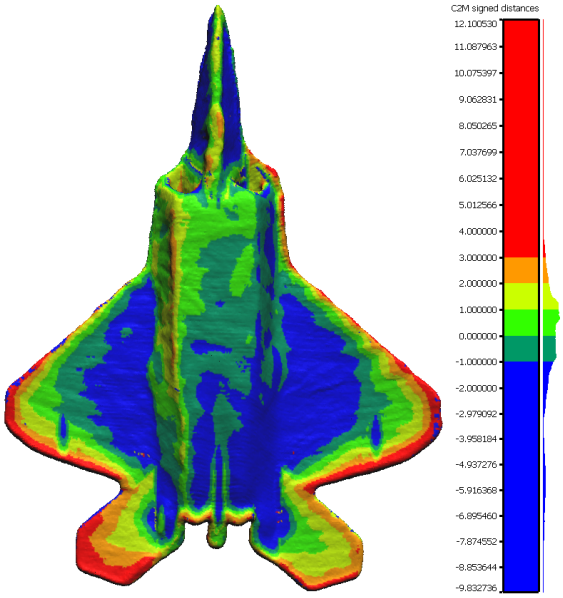


(d) Histogram

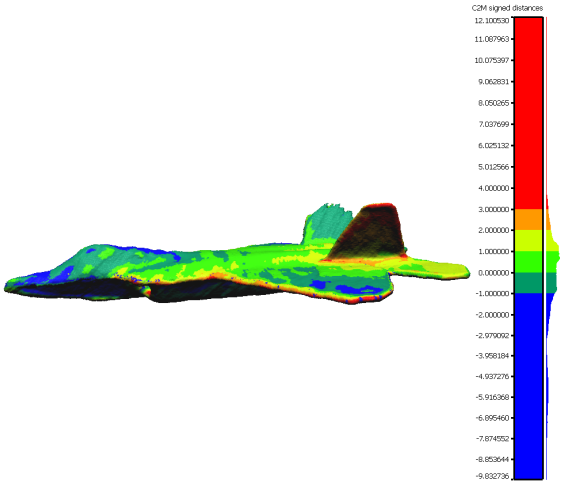
Figure A.14: 1024 vpm at 0.6 Meters.



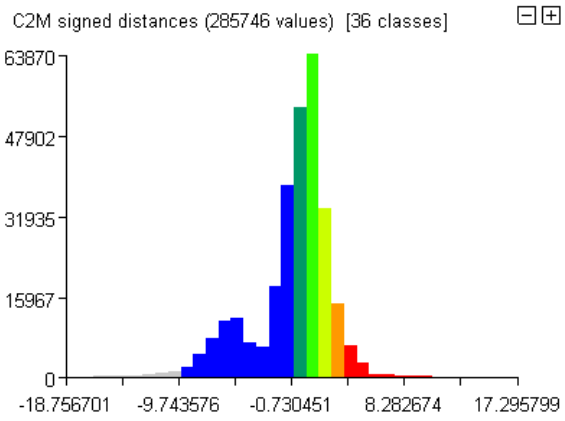
(a) Top View



(b) Bottom View

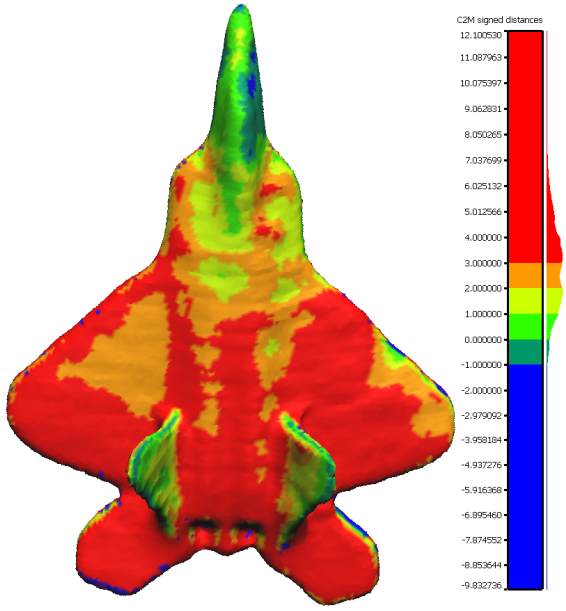


(c) Side View

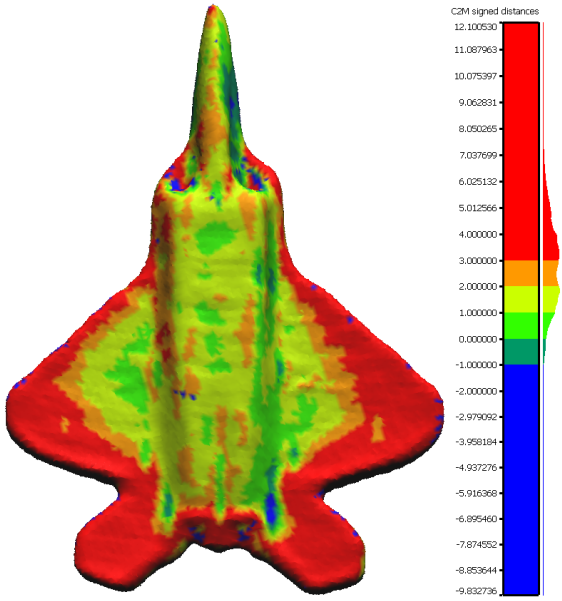


(d) Histogram

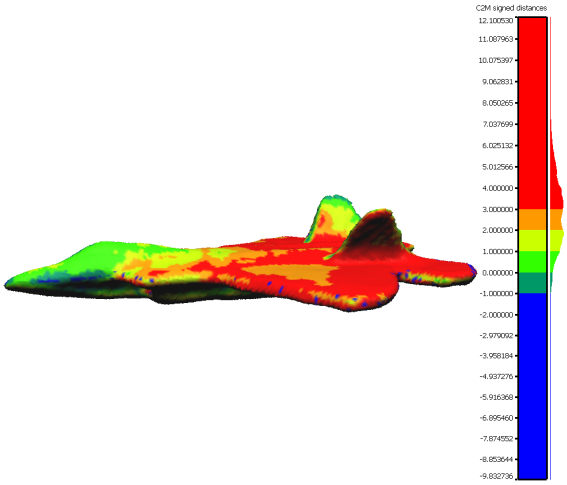
Figure A.15: 1152 vpm at 0.6 Meters.



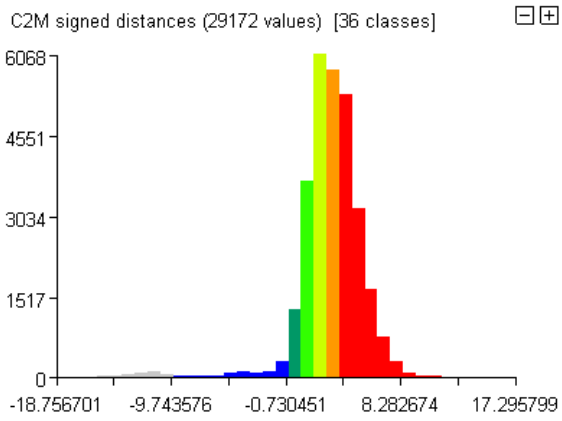
(a) Top View



(b) Bottom View

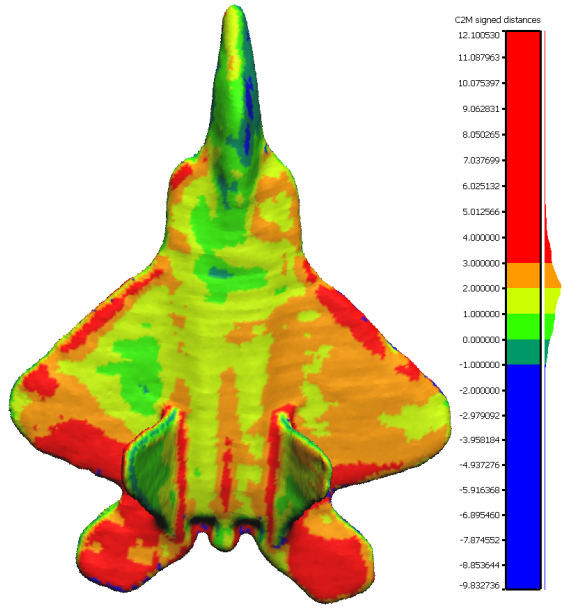


(c) Side View

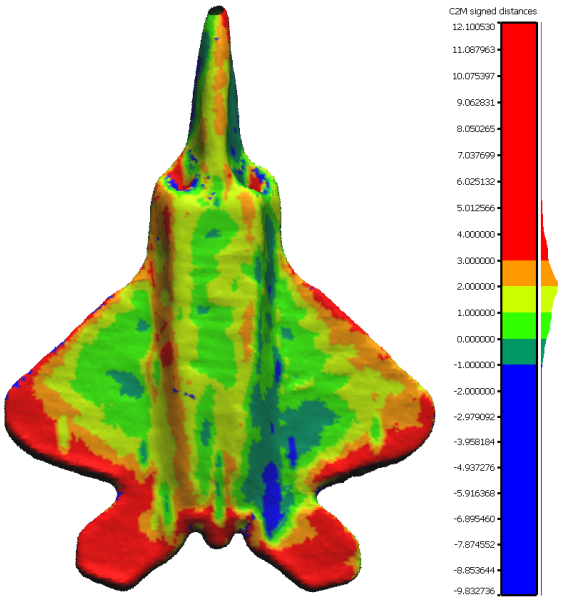


(d) Histogram

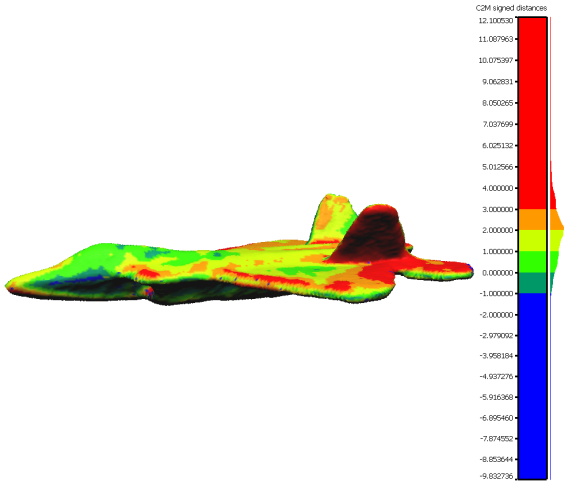
Figure A.16: 384 vpm at 0.8 Meters.



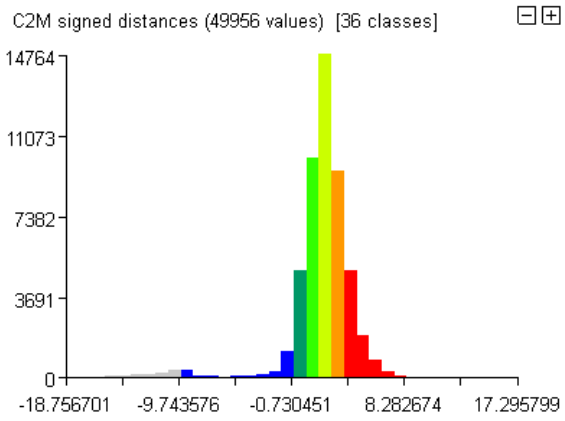
(a) Top View



(b) Bottom View

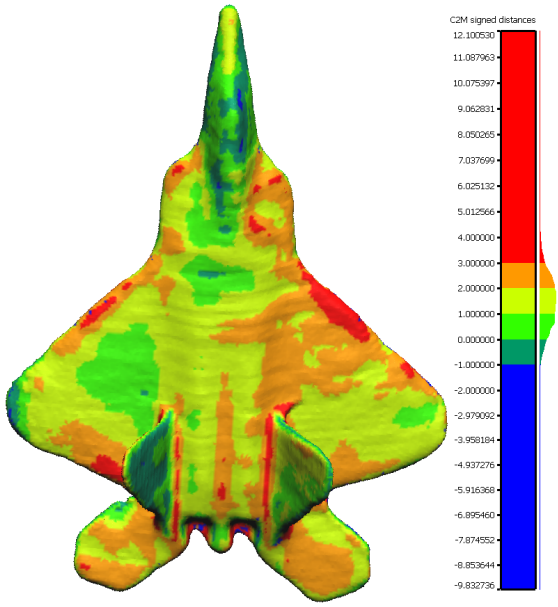


(c) Side View

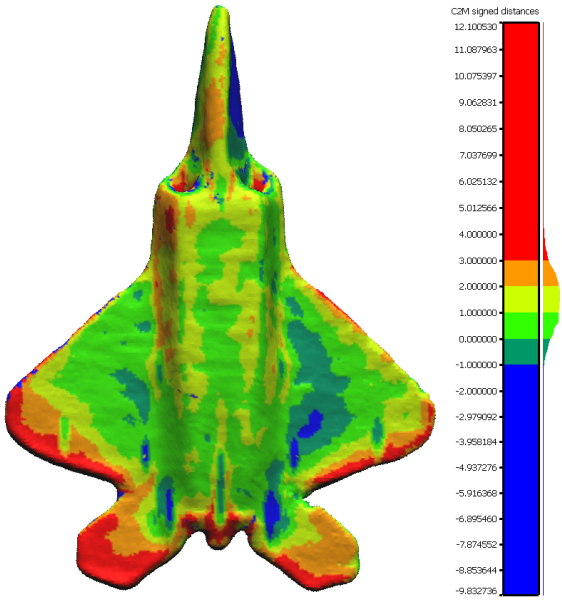


(d) Histogram

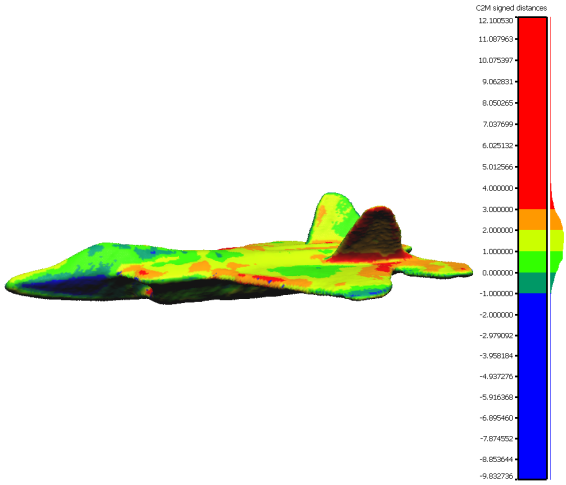
Figure A.17: 512 vpm at 0.8 Meters.



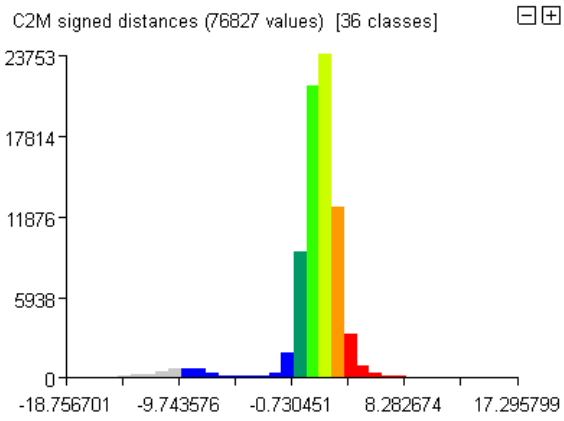
(a) Top View



(b) Bottom View

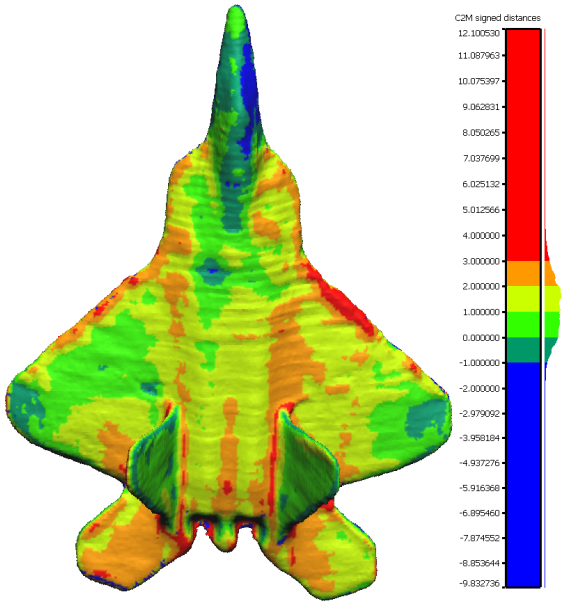


(c) Side View

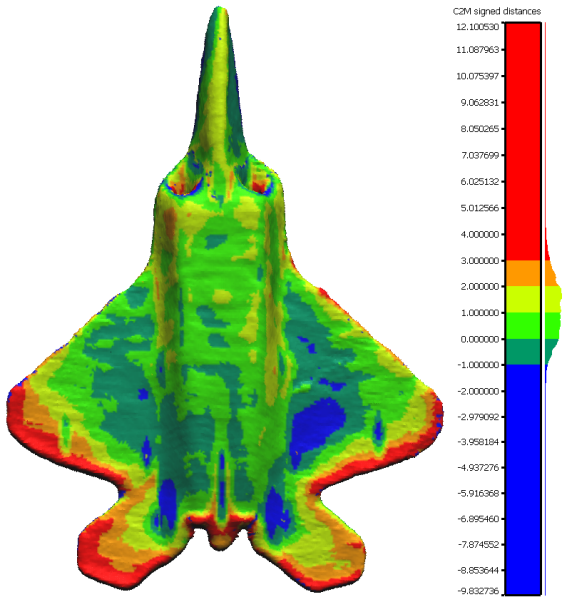


(d) Histogram

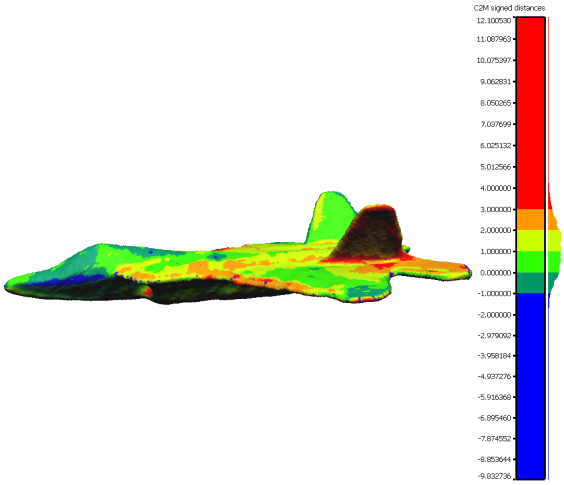
Figure A.18: 640 vpm at 0.8 Meters.



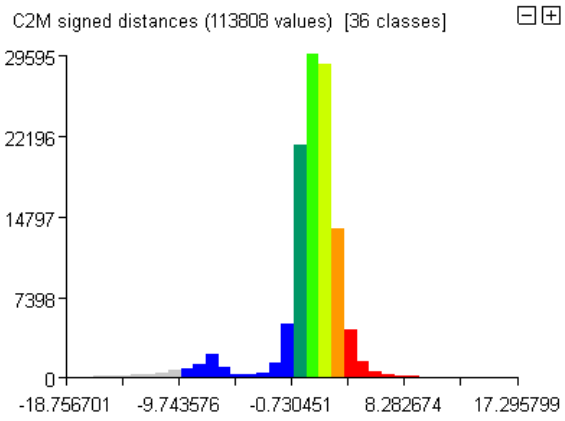
(a) Top View



(b) Bottom View

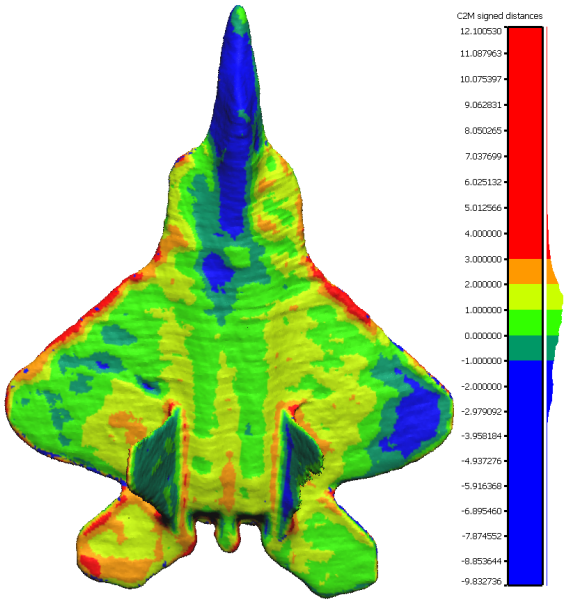


(c) Side View

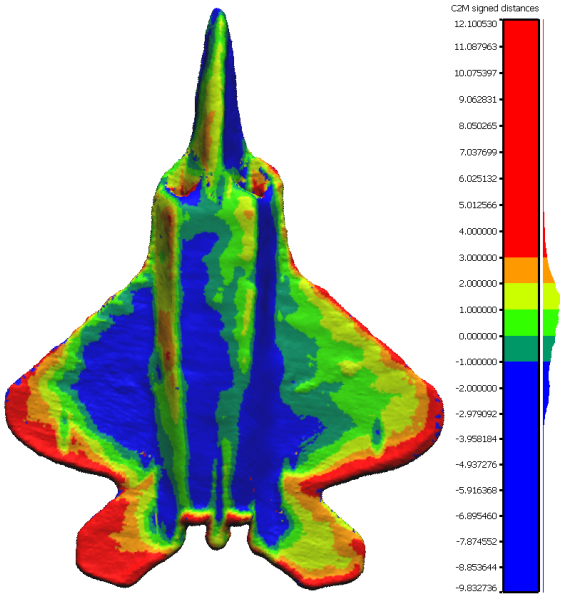


(d) Histogram

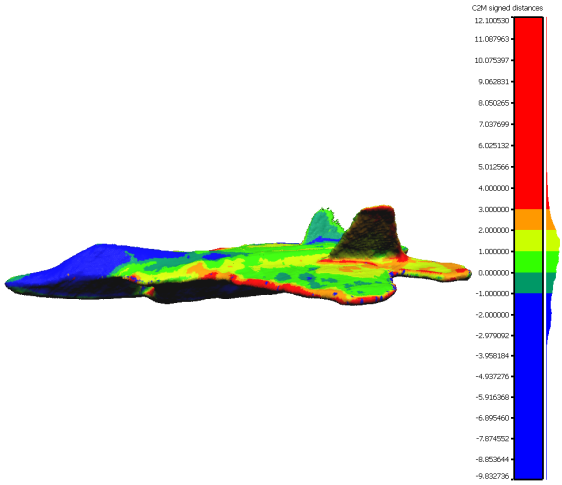
Figure A.19: 768 vpm at 0.8 Meters.



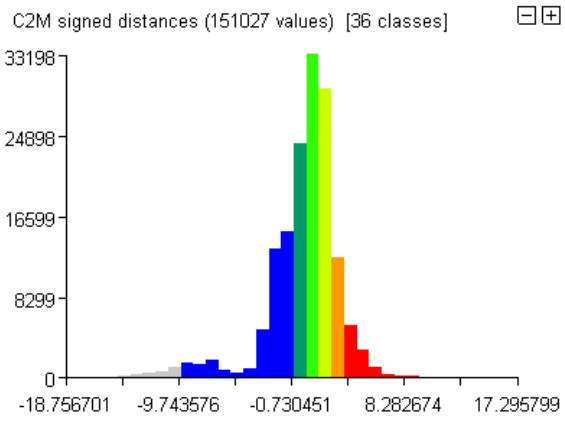
(a) Top View



(b) Bottom View



(c) Side View



(d) Histogram

Figure A.20: 896 vpm at 0.8 Meters.

Bibliography

[1] Cloudcompare wiki, December 2013. URL http://www.danielgm.net/cc/doc/wiki/index.php5?title=Main_Page.

[2] Kinect for windows sdk, September 2013. URL http://msdn.microsoft.com/en-us/library/hh855347.aspx.

[3] Us coin specifications, January 2014. URL http://www.usmint.gov/about_the_mint/index.cfm?action=Coin_specifications.

[4] Stanford computer graphics laboratory, January 2014. URL https://graphics.stanford.edu/data/3Dscanrep/.

[5] Y. Arieli, B. Freedman, M. Machline, and A. Shpunt. Depth mapping using projected patterns, Apr. 3 2012. US Patent 8,150,142.

[6] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. Mesh: Measuring errors between surfaces using the hausdorff distance. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 705–708. IEEE, 2002.

[7] P. J. Besl. Active, optical range imaging sensors. *Machine vision and applications*, 1 (2):127–152, 1988.

[8] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[9] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.

[10] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(4):113, 2013.

[11] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. In *Computer Graphics Forum*, volume 17, pages 167–174. Wiley Online Library, 1998.

[12] FARO. Faro edge and scanarm es features, benefits, and technical specifications, April 2011. URL http://www.faro.com/en-us/products/metrology/measuring-arm-faroarm/overview.

[13] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli. Depth mapping using projected patterns, Apr. 2 2008. US Patent App. 12/522,171.

[14] J. Garcia and Z. Zalevsky. Range mapping using speckle decorrelation, Oct. 7 2008. US Patent 7,433,024.

[15] F. Hausdorff. *Mengenlehre*. Walter de Gruyter Berlin, 1927.

[16] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(9):850–863, 1993.

[17] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047270. URL http://doi.acm.org/10.1145/2047196.2047270.

[18] S. Izadi, R. A. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A. J. Davison, and A. Fitzgibbon. Kinectfusion: real-time dynamic 3d surface reconstruction and interaction. In *ACM SIGGRAPH 2011 Talks*, SIGGRAPH '11, pages 23:1–23:1, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0974-5. doi: 10.1145/2037826.2037857. URL http://doi.acm.org/10.1145/2037826.2037857.

[19] K. Khoshelham. Accuracy analysis of kinect depth data. In *ISPRS workshop laser scanning*, volume 38, page 1, 2011.

[20] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[21] J. E. Marsden and A. Tromba. *Vector calculus*. Macmillan, 2003.

[22] D. Reetz. Futurepicture.org - kinect hacking series, November 2010. URL http://www.futurepicture.org/?p=129.

[23] G. Rote. Computing the minimum hausdorff distance between two point sets on a line under translation. *INFORMATION PROCESSING LETTERS*, 38:123–127, 1991.

[24] H. Roth and M. Vona. Moving volume kinectfusion. In *BMVC*, pages 1–11, 2012.

[25] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

[26] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 438–446. ACM, 2002.

[27] Y. Shirai. Recognition of polyhedrons with a range finder. *Pattern Recognition*, 4(3): 243–250, 1972.

[28] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318. ACM, 1994.

Vita

Captain Jeremy M. Higbee graduated from Prior Lake High School in Prior Lake, Minnesota in 2001. He entered undergraduate studies at the Unversity of Minnesota in Minneapolis, Minnesota, where he graduated with a Bachelor of Science degree in Computer Engineering in May 2005. He was commissioned through Detachment 415 at the University of Minnesota.

His first assignment was at Whiteman AFB as an acquisitions officer in the Research Engineering flight within the 509th Maintenance Operations Squadron in June 2005. In June 2009, he was assigned to the National Air and Space Intelligence Center (NASIC), Wright-Patterson AFB, Ohio where he served as an intelligence analyst. Captain Higbee was then assigned to the Expeditionary Combat Support System (ECSS) Program Office as a Systems Engineer in 2011. In August 2012, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation, he will be assigned to the Space Vehicles Directorate, Air Force Research Labs, Kirtland AFB, New Mexico.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
27–03–2014	Master's Thesis	Sept 2012–Mar 2014

4. TITLE AND SUBTITLE

A Quantification of the 3D modeling capabilities of the KinectFustion Algorithm

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Higbee, Jeremy M., Captain, USAF

5d. PROJECT NUMBER

13-983

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB, OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-14-M-40

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Office of Research & Sponsored Programs
2950 Hobson Way
WPAFB, OH 45433-7765
(937)255-3636 (DSN:785-3636) research@afit.edu

10. SPONSOR/MONITOR'S ACRONYM(S)

AFIT/ENR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In the field of three-dimensional modeling, we continually struggle to quantify how closely the resulting model matches the physical object being represented. When precision measurements are required, they are often left to high-end, industrial systems. The aim of this thesis is to quantify the level of precision that can be obtained from commodity systems such as the Microsoft Kinect paired with the KinectFusion algorithm. Although the Kinect alone is considered a noisy sensor, the KinectFusion algorithm has shown the ability to build detailed surface models through the aggregation of depth information taken from multiple perspectives. This work represents the first rigorous validation of the three-dimensional modeling capabilities of the KinectFusion algorithm. One experiment is performed to measure the effects of key algorithm parameters such as resolution and range, while another is performed to measure the lower bounds at which objects can be detected and accurately modeled. The first experiment found that the KinectFusion algorithm reduced the uncertainty of the Kinect sensor alone from 10 mm to just 1.8 mm. Furthermore, the results of the second experiment demonstrate that the KinectFusion algorithm can detect surface deviations as little as 1.3 mm, but cannot accurately measure the deviation. Such results form an initial quantification of the KinectFusion algorithm, thus providing confidence about when and when not to utilize the KinectFusion algorithm for precision modeling. The hope is that this work will open the door for the algorithm to be used in real-world applications, such as alleviating the tedious visual surface inspections required for USAF aircraft.

15. SUBJECT TERMS

KinectFusion, Kinect, 3D Modeling

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maj. Brian G. Woolley (ENG)
U	U	U	UU	117	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4618 Brian.Woolley@afit.edu